Л.А. РАСТРИГИН

С КОМПЬЮТЕРОМ НАЕДИНЕ

С КОМПЬЮТЕРОМ НАЕДИНЕ

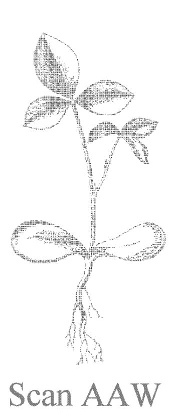


Основана в 1947 году

Выпуск 1157

Л. А. РАСТРИГИН

С КОМПЬЮТЕРОМ НАЕДИНЕ





Москва
«Радио и связь»
1990

ББК 32.97
Р24
УДК 681.3.015.001.85

Рецензент: д-р техн. наук, проф. Д. А. Поспелов

Редакция литературы по информатике вычислительной технике

Растригин Л. А.

Р24 С компьютером наедине.—М.: Радио и связь, 1990.— 224 с.: ил.—(Массовая радиобиблиотека: Вып. 1157)

ISBN 5-256-00441-1.

Посвящена диалогу с компьютером и правилам взаимодействия с ним, знакомит с возможностями современных компьютеров. Читатель, который хочет использовать компьютер для решения своих задач, найдет необходимые сведения о том, как работает компьютер, как составляются программы на различных языках программирования, об искусственном интеллекте, экспертных системах и о многом другом. А помогут ему в этом комиссар Мегрэ и его помощник Поль, страстный поклонник компьютеров.

Для радиолюбителей, может быть полезна широкому кругу читателей.

Р $\frac{2404000000—034}{046(01)—90}$ 135—90 **ББК 32.97**

ISBN 5-256-00441-1

ДЕТЕКТИВ ВМЕСТО ПРЕДИСЛОВИЯ

...Сигарета обожгла пальцы. Мегрэ вздрогнул, поморщился и с досадой стряхнул пепел с пиджака. «Хорошо, что еще не прожёг пиджак,— подумал он.—Досталось бы мне дома». Очень хотелось отвлечься от дела, но отвлечься не удавалось. «Старею, видно»,— невесело подумал Мегрэ.

А началось все довольно тривиально. И вообще во всей этой истории с самого начала все было как-то просто, обыденно.

Звонок в комиссариат: «Труп в гостинице». Осмотр места происшествия подтвердил возникшее подозрение о тривиальности преступления. Отравилась или была отравлена молодая девушка.

Она долго жила в этой гостинице и много задолжала. К ней часто приходил молодой человек обыкновенной наружности, без примет, молодость была его самой яркой чертой. Обстоятельства осмотра тоже были самыми обыкновенными: испуганный директор гостиницы, умоляющий ничего не сообщать репортерам, настороженные взгляды горничных, хорошо понимающих, что допрос начнется с них, любопытные лица постояльцев и сонные глаза врача-эксперта и фотографа, которых, казалось, уже ничего взволновать не могло.

Необычными были лишь огромные глаза Жака, молодого стажера из полицейской школы, которого только что прикрепили к группе Мегрэ. Он с трудом отвел их от открытого флакона, зажатого в руках погибшей девушки. Легкий запах горького миндаля подсказывал стандартное решение — цианистый калий, которое наверняка можно было бы и не проверять в химической лаборатории. Но Поль, помощник комиссара, осторожно вынул флакон из застывших пальцев, взглянул на отпечатки и вложил в зажим своего чемодана «последней помощи», как его презрительно называл врач.

Все шло своим чередом. Поль заканчивал осмотр места происшествия, Жак начал опрос свидетелей, Сид запрашивал данные по телефону — все были заняты своим делом. Мегрэ рассеянно оглядывал комнату. Ничего необычного, за что можно зацепиться, что могло бы стать ключом к расследованию, он не заметил. «Надо это дело поручить Полю. Пора ему действовать самостоятельно. И начинать надо с простого дела. Он быстро вытрясет из безликого поклонника необходимые сведения и решит что же это, самоубийство или нет. Основания будут веские»,— подумал Мегрэ, зная, как тщательно работает Поль.

Обыкновенность ситуации успокаивала, убаюкивала. «Наконец сегодня приду домой вовремя»,— размечтался Мегрэ. И вдруг он увидел ЭТО. Звук для него исчез. Как в немом кино, он видел, что его сотрудники говорят о чем-то, что-то спрашивают у понятых, но ничего не слышал. Видел только ЭТО.

Сначала Мегрэ удивился, что ЭТОГО никто не увидел. Даже пунктуальный

Поль едва ли занес ЭТО в список осмотра комнаты. Но сразу понял, что никто из присутствующих, кроме него, не может заметить ЭТО. Ведь только он один знает, что ЭТО. Мегрэ похолодел, ведь вызов могла принять и другая группа. И тогда ЭТО никто бы не заметил. И преступление (он уже не сомневался в том, что это преступление) так и осталось бы нераскрытым. Да и не только это преступление, но и все следующие, которые неизбежно произойдут благодаря ЭТОМУ. Он хорошо знал, какую «дичь» ухватил. Несчастная жертва, которую уже выносили санитары была даже не пешка, а паутинка в делах международных гангстерских банд. Но, потянув за эту паутинку, он сможет вытянуть таких китов мировой мафии, которые и не снились Интерполу.

— Поль!— резко прервал молчание Мегрэ.— Немедленно займись этим молодчиком, а я наведу кое-какие справки.

— Слушаю, шеф!— ответил Поль.— Только закончу осмотр. Не думаю, чтобы этот парень успел далеко скрыться, даже если он замешан в этом деле.

— Осмотр пусть продолжит Жак. Пора ему на практике проверить знания, полученные в полицейской школе,— сухо сказал Мегрэ и стал осторожно заворачивать ЭТО в свой носовой платок.

— Слушаю, мэтр!— торопливо сказал Жак и испуганно покосился на Поля, который укоризненно наблюдал, как Мегрэ положил в карман какую-то пустяковину.

Следующие два часа Мегрэ провел в телетайпной своего комиссариата, посылал запросы о лицах, компаниях, событиях и преступлениях своим коллегам в разных частях света — в Лондон, Чикаго, Мехико, Карачи, Рио-де-Жанейро, Токио, Сингапур и другие центры борьбы с мафией. Память комиссара работала четко и точно:

— Лейтенанту Нгаби. Министерство внутренних дел. Найроби.—диктовал Мегрэ.— Старина, сообщите срочно, проходил ли по делу убийства президента страховой компании «Космос» китаец Ли Ван Чу из Гонконга. Его клички есть в справочнике Интерпола.

— Майору Пронину. Москва. Петровка, 38. Отдел по борьбе с хищениями.— Коллега, прошу срочно сообщить о связях компании, которая пыталась провести через Москву груз наркотиков из Афганистана в ФРГ транзитом.

— Инспектору Синг Рао. Калькутта. Индия. Отдел по борьбе с бандитизмом.— Подтвердите...

— Сержанту Полу Джонсону. Инспекция по борьбе с наркоманией. Мельбурн, Австралия.—Прошу сообщить... И т. д. и т. п.

Комиссар запрашивал все, что могло иметь отношение к ЭТОМУ. Пусть его коллеги сообщат ему побольше, а он уже сумеет выбрать жемчужины, чтобы из них собрать ожерелье неопровержимых доказательств и покончить с главарями мафии. Они-то считают себя в полной безопасности.

Вскоре начали поступать ответы. С телетайпов потянулись широкие ленты бумаги. Когда он пришел в свой кабинет, то увидел, что буквально все завалено телетайпными сообщениями. Сержант подносил все новые и новые, а телетайпы продолжали работать. И так будет до завтрашнего вечера,— смекнул Мегрэ, зная, как поступают ответы на запросы, направленные в дальние точки планеты. И он начал читать сообщения.

...И вот глубокой ночью, прочитав первую сотню сообщений, Мегрэ понял, в ка-

кую трудную ситуацию он попал. Из моря получаемой информации ему предстояло «выудить» несколько убедительных фактов, которые подтвердили бы его версию. Он не знал, какие именно факты, но сразу определил бы их, как только увидел. Все было бы хорошо, если бы не такое обилие сообщений. Жемчужины информации так и лежали в своих раковинах. Мегрэ хорошо понимал, почему поток столь обилен. Зарубежные коллеги старались сообщить побольше, справедливо считая, что из большего объёма сведений можно выявить больше информации, чем из малого. Думали ли они, что этот объем будет ему не под силу переработать. А равнодушные посылали сообщения без особого разбора — лишь бы отписаться. Но и те, и другие одинаково мало помогли и одинаково усложнили задачу комиссара. Их нельзя было судить строго, ведь конечная цель им была неизвестна!

— Пропустить бы все это через какое-то сито,— невесело подумал Мегрэ и вспомнил сказку о волшебной мельнице, которая перемалывала все, что угодно, а выдавала лишь то, что заказывал ее хозяин. Помнится тогда, в детстве, он был очень удивлен, что ее злой хозяин заказал мельнице золото, а не ореховое пирожное. Вот и сейчас мне нужно не золото, а ...информация. Заложил бы в мельницу всю эту бумагу и ...— тут Мегрэ насупился.— Черт знает, какие мысли лезут в голову. Это все от собственного бессилия, неумения, незнания. Стоп! Нужно действовать по плану, только так можно решить трудную проблему. Что я не умею и не знаю? Не умею справиться с этим морем сообщений. Не знаю, как это сделать. А кто знает? Тот, кто занимается переработкой информации! И Мегрэ поздравил себя с этим «открытием». Это был тупик, вывести из которого могло бы или чудо, или ...

— Поль!— позвал Мегрэ по селектору. Он не сомневался, что его верный помощник находится где-то рядом. Так было всегда в трудные минуты.

— Слушаю, шеф,— немедленно отозвался Поль.— Я иду к Вам.— Он хорошо знал комиссара и понимал его с полуслова.

— Малыш!— отечески обратился Мегрэ к долговязому Полю, когда тот появился в дверях.— Видишь эти сообщения? Постарайся, пожалуйста, переработать каким-то образом все это, чтобы самое главное разместилось не более чем на десятке страниц машинописного текста. Это связано с нашим последним делом.— И отвернулся, чтобы скрыть улыбку. К его удивлению, Поль даже не улыбнулся и стал собирать сообщения.

— Что намерены делать?— по-деловому спросил Мегрэ.

— Отослать все это профессору Мариаку в Гренобль.— отрапортовал Поль.— Ведь он занимается переработкой информации с помощью вычислительных машин. Помните дело об ограблении банка, которое он помог нам раскрыть за два часа?

Мегрэ хорошо помнил это дело*. Тогда надо было определить особенности фигур людей по видеозаписи ограбления. Для этого понадобилась одновременная совместная работа тысяч компьютеров, объединенных в вычислительную сеть, раскинувшуюся на двух континентах. Только такая сеть могла за два часа идентифицировать гангстеров. Мегрэ не без гордости вспоминал это дело. Помнится, министр даже упомянул его в годовом отчете. А уж газетчики раздули дело до безобразия, что дало повод для довольно ехидных шуток коллег. Но апофеозом был

* «Это дело» комиссара описано в книге автора «Вычислительные машины, системы, сети...» (М.: Наука, 1982).

визит очаровательной Мадо, секретарши префекта. Она обратилась к Мегрэ с просьбой «вычислить», с кем путается ее Жорж, и даже притащила с собой в сумочке карманный калькулятор. С тех пор упоминание о компьютере в группе Мегрэ считалось неприличным — слишком быстро «заводился» шеф.

Глядя в ясные глаза Поля, Мегрэ понял, что это не розыгрыш, и возмущенно с горячностью воскликнул:

— Но ведь в том деле компьютерная сеть вычисляла, занималась тем, для чего она и создана,— для вычислений.

— А почему Вы думаете, что любая обработка информации не может быть сведена к вычислениям?— спросил Поль.

— Знаю я Вас, молодежь,— по любому поводу лезете с вопросами к компьютеру, а все ваши вопросы одинаковы: где достать да с кем познакомиться!— брюзжал Мегрэ.

— Но, шеф, Вас ведь тоже интересует, где и как достать факты по делу отравленной девушки,— возразил Поль.

— Да, но я...— Мегрэ поперхнулся, задумался и махнул рукой.— Валяйте, Поль, отсылайте материалы Мариаку, пусть помаракует с ними. Компьютер не более чем инструмент в руках человека, даже если это сыщик. Не правда ли, малыш?

Тут настал черед Поля. Последние слова комиссара были отголосками старого спора между ними — сможет ли компьютер заменить человека в розыскной работе. У Поля на этот счет была своя точка зрения — полная компьютеризация сыскного дела. Не люди, а автоматы должны заниматься розыском — таков был девиз Поля. Люди нужны лишь для ремонта автоматов и обеспечения их необходимой информацией, если они сами не смогут это сделать. Легко представить, как принимал эту «доктрину» Мегрэ.

— В чем-то Поль, несомненно, прав,— размышлял Мегрэ.—Но несомненно и то, что не все доступно компьютеру. Как, например, он бы мог заметить ЭТО? Здесь уже интуиция, опыт. А может быть, интуиция — это тоже всего лишь «переработка информации»?— испуганно подумал Мегрэ и горестно вздохнул.— Возможно, когда-нибудь это так и будет, а пока будем твердо стоять на своих ногах. Я могу делать то, что не может сделать сейчас ни один компьютер. Да, ни один,— упрямо подумал Мегрэ.— Но компьютер и сейчас тоже может сделать то, что не могу сделать я. Что из этого следует? Нам нужно объединиться — создать нечто вроде симбиоза человека и машины, в котором носителем опыта, интуиции и ассоциаций будет человек, а на компьютер будет возложена функция переработки информации по правилам, предложенным его программами. И не нужно противопоставлять меня компьютеру,— подумал Мегрэ,— меня нужно дополнить компьютером, а не компьютер мной в качестве масленщика. Он решительно против тезиса Поля о роли человека в будущем мире автоматов.

— О чем это вы, шеф?— спросил Поль.

— Поль,— замялся Мегрэ,— не дашь ли ты мне какую-нибудь книжку о компьютерах? Но только не о том, как они устроены. А о том, как происходит эта самая «переработка информации» и какие у него возможности, чтобы можно было ими воспользоваться в нашей сыскной работе.

— Давно пора, шеф,— улыбнулся Поль.— Вот недавно вышла в Москве книжка,— и Поль вытащил из кармана книгу (эта книга перед тобой, дорогой читатель).— Здесь все описано достаточно полно и просто.

— А автор тоже масленщик,— настороженно спросил Мегрэ,— и считает, что будущее человечества только в обслуживании компьютеров?

— Да нет, шеф. Это разумный парень, хотя и не без академических загибов, видно, занимается наукой. Но для первого знакомства с возможностями компьютера этого вполне достаточно.

— «С компьютером наедине». А что значит это название?

— Эта книга о диалоге с компьютером,— заметил Поль.

— Диалог с компьютером,— задумчиво повторил Мегрэ,— это лучше, чем «симбиоз». Да, именно д и а л о г, где каждый выступает в своей роли, со своим мнением, своими возможностями. Я согласен на диалог с компьютером, Поль, но никогда не буду его масленщиком.

— От Вас этого и не потребуется,— улыбнулся Поль.—Дай бог ему ответить хоть на малую толику вопросов, которыми Вы его завалите.

— До свиданья, Поль,— Мегрэ засунул книгу в портфель.—Не забудьте заправить к утру вашу масленку.

— До свиданья, шеф, буду помнить,— и Поль влюбленными глазами проводил сутулую фигуру комиссара.

I. КОМПЬЮТЕР — ВАШ СОБЕСЕДНИК

1. ЗНАКОМЬТЕСЬ — КОМПЬЮТЕР

РОЖДЕНИЕ КОМПЬЮТЕРА

Компьютер, он же электронная вычислительная машина (ЭВМ),— детище века, появился в ответ на вполне конкретные потребности человечества — нужно было быстро и много вычислять. С этой нехитрой задачей человек справлялся, но слишком медленно. Механические приспособления вроде конторских счет или арифмометров не решали проблему. Нужно было совершать тысячи и миллионы арифметических операций в секунду. Механика здесь бессильна. Помогла электроника.

И вот в 1945 г. появился первый компьютер. Назвали его ENIAC (ЭНИАК). Создан он был для расчета баллистических таблиц. Дело в том, что для точной стрельбы из артиллерийских орудий нужно учитывать очень много факторов: скорость ветра, изношенность ствола орудия, его температуру, массу и тип снаряда, вид пороха и многое другое. Чтобы определить направление ствола орудия, нужно предварительно сделать очень много вычислений. А так как в бою времени нет, то для этого обычно пользовались таблицами. Для каждого нового типа орудия требовались свои таблицы, составлять их приходилось годами: этим занимались люди, вооруженные лишь счетами и арифмометрами. Это задерживало использование нового орудия. Так возникла важная задача быстрого составления баллистических таблиц. Именно для ее решения и был создан в США первый компьютер, открывший дорогу компьютерному веку.

Компьютеры стали появляться десятками в год, потом сотнями, тысячами... и, наконец, миллионами. Менялся не только их облик и содержание, но и функции.

НЕ СЧЕТОМ ЕДИНЫМ...

Первой функцией, ради выполнения которой и был создан компьютер, была вычислительная. И ENIAC был автоматом для вычислений и только вычислений. Но ...как всякая хорошая разработка, он оказался пригодным и для других незапланированных целей. Одна из таких новых и очень полезных функций — невычислительная. Любой компьютер может не только вычислять, но и хранить информацию, более того, может преобразовывать

ее к требуемому виду. Все эти явно невычислительные функции реализуются на современном компьютере вместе с вычислительными, одним и тем же устройством. Это и позволяет называть компьютер не только вычислительным устройством (или ЭВМ), но и устройством обработки информации. Именно эта обработка объединяет и вычислительные и невычислительные функции компьютера. Действительно, всякий счет является обработкой информации, но представленной в специальной числовой форме. Вычисления являются лишь частным случаем обработки знаковой информации — информации, представленной в знаковой форме, например в виде чисел, букв, слов, иероглифов и любых других знаков, имеющих определенный смысл.

Как легко заметить, вычисления составляют очень малую долю в общем процессе обработки информации. Это обстоятельство и привело к тому, что компьютеры из вычислительных машин постепенно стали превращаться в машины обработки информации (в широком смысле). И сейчас вычисления занимают лишь 10% общего компьютерного времени. Остальные 90% приходятся на обработку нечисловой информации — реализуется невычислительная функция компьютера. Это, прежде всего, поиск информации в компьютерных библиотеках (банках данных), моделирование поведения сложных систем, обработка изображений и многое другое. Вот и получилось, что выполнение именно этих невычислительных функций занимает львиную долю общего машинного времени современного компьютера. И эта доля продолжает увеличиваться. Так что современные ЭВМ лишь в очень малой степени (лишь на 10 %) можно назвать вычислительными. Возможно, поэтому в последнее время у нас вместо слова ЭВМ все чаще используют слово «компьютер», хотя в переводе с английского оно означает все тот же «вычислитель». Но не будем нарушать сорокалетнюю традицию и сохраним добрые старые имена ЭВМ и компьютер за сложными программируемыми электронными автоматами обработки информации — именно так определяется их огромная область применения.

Здесь Мегрэ удовлетворенно хмыкнул — он был уверен, что компьютеры только вычисляют, и поэтому так мало интересовался ими, хотя и понимал важность вычислений для экспертизы. Действительно, без вычислений не ответишь на вопрос: одно и то же или разные лица изображены на двух фотографиях, или по двум найденным пулям не определишь, в каком порядке были произведены выстрелы, и не решишь многие другие задачи, решаемые экспертами, вооруженными современными компьютерами.

Но одно дело, когда следователь ставит четкий вопрос перед экспертизой и даже дает возможные варианты ответа («он» или «не он»), и совсем другое дело поставить вопрос. Это значительно труднее, чем ответить на него (хотя, как говорят, один дурак может задать столько вопросов, что на них не ответят и десять мудре-

цов). Ведь недаром говорят, что правильная постановка задачи (а это и есть хороший вопрос) в значительной степени определяет ее решение.

Так что правильно, что компьютеры нагружают невычислительными задачами,— подумал Мегрэ.— Именно такие задачи приходится решать сыщику. Но здесь нужна логика и еще раз логика! А владеет ли ею компьютер?

ЛОГИЧЕСКИЕ АТОМЫ КОМПЬЮТЕРА

Итак, компьютер — очень сложный электронный автомат, не более. Этот электронный автомат и обрабатывает вводимую в него информацию. Сигналы, циркулирующие внутри компьютера в процессе такой обработки, очень просты. Их всего два: 0 и 1. Это не числа, а *имена* сигналов, их можно было бы назвать А и В, или Иван и Марья, или красное и черное — любыми, но разными именами. Так устроена *двоичная система*, которая обходится лишь двумя знаками для кодирования информации, циркулирующей в компьютере. Здесь 0 и 1 — условные *логические значения* сигналов, которые не следует путать с их физическим содержанием. А физически 0 может соответствовать низкому напряжению, а 1— высокому (или наоборот по договоренности) или 1 может соответствовать наличию импульса электрического или светового, а 0— его отсутствию и т. д. Способов представления 0 и 1 в компьютере много, но используется обычно лишь один для данного типа компьютера.

Итак, лишь два вида сигналов используются в современном компьютере. Именно поэтому говорят что компьютер работает по принципу «или все, или ничего» (или 1, или 0), третьего не дано. Этот принцип обеспечивает преимущества компьютеру перед другими автоматами, где число значений, принимаемых его сигналами, больше двух.

В истории компьютерной техники были разработки, где использовалось более двух сигналов: три —1, 0 и 1 (троичная система) и даже 10 (десятичная). Но, как показал опыт, они не получили дальнейшего развития, и сейчас используется лишь двоичная система сигналов. Этому есть две причины.

Во-первых, надежность. Так, случайной помехе труднее изменить 0 на 1 (и наоборот) в двоичной системе, чем 0 на —1, 0 на 1 (и наоборот) в троичной системе. Дело в том, что интервал от 0 до 1 в двоичной системе больше, чем интервал от —1 до 0 или от 0 до 1 в троичной, так как в последней —1 соответствует, например, низкому напряжению, 1— высокому (это 0 и 1 в двоичной системе), а 0— промежуточному напряжению, которое, естественно, легче изменяется помехой на высокое (1) или низкое (—1).

Другим существенным преимуществом двоичной системы представления информации является то, что для обработки потоков двоичной информации не нужно иметь много преобразователей: вариантов таких преобразований может быть немного. Каждый

преобразователь преобразует несколько потоков информации (потоков нулей и единиц) в один поток. Если не задумываться о том, какой физический процесс при этом используется, то следует говорить о логических преобразователях, или, как их чаще называют, *логических элементах*. Таких элементов для двоичной системы сигналов немного. Это, прежде всего, элементы, названные условно ИЛИ, И, НЕ. Эти три логических элемента и являются основными логическими «атомами», из которых состоит компьютер. Именно с их помощью осуществляется та самая обработка информации, для которой и создан любой компьютер.

Однако одних логических функций мало для работы компьютера. Одной из важнейших его функций является *запоминание*. Компьютер должен сохранять вводимую в него информацию, а также все промежуточные и окончательные результаты обработки. Причем запоминать компьютеру нужно все те же логические нули и единицы, которые циркулируют в нем. Проще всего это осуществляется с помощью магнитных элементов памяти. Намагничивание в одном направлении элементарного магнита означает запоминание 0, а в другом —1 или наоборот. Так устроена внутренняя память компьютера. Сколько магнитиков, столько и единиц информации способен запомнить компьютер. Но процесс перемагничивания даже самых маленьких магнитиков требует времени и, следовательно, снижает производительность компьютера. Самая же быстрая память построена на логических элементах.

Логические элементы могут быть реализованы различным образом. Если используются электронные элементы (транзисторы, резисторы, емкости и т. д.), то компьютер называют электронным или электронной вычислительной машиной (ЭВМ). При использовании оптоэлектронных элементов, сочетающих и оптические и электронные методы переработки информации, компьютер называют оптоэлектронным. Сейчас начинается освоение оптоэлектронной техники, и вскоре такие компьютеры станет выпускать промышленность. А при использовании биологических логических элементов компьютер называют биокомпьютером, пока только ведутся интенсивные исследования в этом направлении. Как видно, на логическом уровне совершенно не важно, на какой элементной базе (электронной, оптической или биологической) построен компьютер. Именно поэтому так удобно рассматривать логическую схему компьютера. Ею мы и будем заниматься в дальнейшем.

ПРОГРАММИРУЕМОСТЬ — ОСНОВНОЕ СВОЙСТВО КОМПЬЮТЕРА

Итак, компьютер — автомат для переработки информации. И в этом смысле он сродни первому автомату — автомату Герона, созданному им до нашей эры для торговли «святой водой». Этот автомат — прообраз автомата для торговли газированной водой, которым наверняка приходилось пользоваться каждому из

нас. Такой автомат преобразует информацию о достоинствах монеты в команду на выдачу стакана (200 г.) газировки с сиропом, если опущена трехкопеечная монета, и без сиропа, если однокопеечная. Компьютер в отличие от обычного автомата (например, автомата для продажи газированной воды) может решать не одну, а множество задач и даже одновременно в зависимости от программы. Компьютер — это *программируемый автомат*. Именно в этом его отличие и достоинство.

Изменяя программу, мы перестраиваем компьютер на решение других задач. Например, на одном и том же компьютере можно вычислять и баллистические таблицы (вспомним ENIAC) и рассчитывать оптимальный рацион кормления животных. Но для каждой задачи нужна своя программа.

Что же такое компьютерная программа? Это точная инструкция компьютеру, как обрабатывать исходные данные, чтобы получить требуемый результат. Например, для решения квадратного уравнения $ax^2+bx+c=0$ необходимо ввести в компьютер исходные данные — значения коэффициентов a, b, c, а также программу вычисления корней этого уравнения по исходным данным. В соответствии с этой программой компьютер реализует процедуру вычисления известной формулы, подставляя в нее исходные данные, т. е. действует, по сути, так же, как аккуратный школьник.

При обращении к информации, хранимой в памяти компьютера, исходными данными является запрос, т. е. что же именно интересует вас. Программа поиска нужной информации уже имеется и хранится в той же памяти компьютера. В этом случае мы имеем дело с информационно-поисковой системой. Она образуется компьютером, в памяти которого хранятся информационные массивы и программа для поиска в этих массивах той информации, которая описана в запросе. Таким образом, и данные, и программа должны быть в памяти компьютера. Если чего-либо из этого комплекта недостает, то решить поставленную задачу компьютер не сможет, как, впрочем, и человек, попавший в такую же ситуацию.

ДВОИЧНОЕ «МЫШЛЕНИЕ» КОМПЬЮТЕРА

Мы уже говорили, что компьютер манипулирует информацией, представленной в двоичной форме — в виде нулей и единиц. Об этом стоит поговорить подробнее.

Как же устроена двоичная система? Да так же, как и привычная нам десятичная, только в основу положена не десятка, а двойка. В десятичной системе любое целое n-значное число можно представить в виде

$$(a_n a_{n-1} \ldots a_1 a_0)_{10} = a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \ldots + a_1 \cdot 10^1 + a_0 \cdot 10^0 \cdot a_i$$

Здесь индекс 10 означает, что число десятичное, а цифры могут

принимать любые значения от 0 до 9. Совершенно аналогично представление двоичного числа:

$$(b_m b_{m-1} \ldots b_1 b_0)_2 = b_m \cdot 2^m + b_{m-1} \cdot 2^{m-1} + \ldots + b_1 \cdot 2^1 + b_0 \cdot 2^0,$$

где индекс 2 означает двоичное представление этого числа, а числа b_i могут быть 0 или 1. Например, $(10011)_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = (19)_{10}$.

Дробные числа образуются аналогично:

$$(b_m \ldots b_0, b_1 \ldots b_{-n})_2 = b_m \cdot 2^m + \ldots + b_0 \cdot 2^0 + b_{-1} \cdot 2^{-1} + \ldots + b_{-n} \cdot 2^{-n},$$

где коэффициенты с отрицательными индексами описывают дробную часть числа. Например, $(101{,}011)_2 = 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2^2} + 1 \cdot \frac{1}{2^3} = 5{,}375)_{10}$.

Преимущество двоичного представления чисел состоит в том, что все операции с ними реализуются очень простыми аппаратными средствами — элементами И, ИЛИ, НЕ. В этом секрет успеха двоичного счета.

Вся перерабатываемая компьютером информация (числа, текст, рисунки и графики) должна быть сначала представлена в двоичной форме — в виде наборов нулей и единиц, с которыми и работает компьютер. Так, для обработки изображения его представляют в виде отдельных точек, расположенных определенным образом, а прямоугольное изображение можно представить в виде таблицы. Каждый элемент этой таблицы определяет яркость соответствующей точки, записанной в виде двоичного числа. Так, для обработки изображения телеэкрана размером 800×625 точек (около полумиллиона) в память компьютера нужно ввести таблицу из 800×625 двоичных чисел, а для цветного изображения — три таких таблицы для яркостей красного, зеленого и синего цветов.

Единицей информации и ее минимальной порцией является *бит* — один разряд двоичного числа. С помощью n двоичных чисел можно закодировать 2^n различных сигналов, таких, как обычное десятичное число (например, 123,876 или $125 \cdot 10^{-14}$), слово (например, «Москва») или команду (например, «сложить»). Способов такого кодирования можно придумать много. В различных ЭВМ используют различные способы кодирования. Но бит — слишком малая единица информации. И поэтому используется байт или, точнее, октет — это 8-разрядное двоичное число, с помощью которого можно закодировать $2^8 = 256$ различных сигналов. Их хватает, чтобы обозначить все цифры, буквы латинского и русского алфавитов, прописные и строчные, и специальные знаки, включая знаки пунктуации. Так, например, в стандарте ЕС ЭВМ (так называют Единую Серию ЭВМ) число 1987 записывается в виде

1	9	8	7
11110001	11111001	11111000	11110111

а слово «Москва» так:

М	О	С	К	В	А
11010100	11010110	11000011	11010010	11000010	11000001

Есть еще более крупная информационная единица — машинное *слово*. Его длина кратна чаще всего байту, т. е. 1, 2, 4 и 8 байт или 8, 16, 24 и 64 бит. Переработка информации в ЭВМ происходит только такими словами. Это значит, что все разряды машинного слова обрабатываются одновременно.

Мощные ЭВМ работают с 8-байтовыми (64-битовыми) словами, а самые простые ЭВМ с однобайтовыми. Это вовсе не значит, что ЭВМ с однобайтовыми словами не сможет делать точных вычислений, просто ей придется перерабатывать длинное число в несколько этапов, а значит, дольше.

ИЗ ИСТОРИИ КОМПЬЮТЕРА

Первые современные ЭВМ появились только в середине 40-х годов нашего века. Тогда же знаменитый Джон фон Нейман (1903—1957) изложил принципы построения ЭВМ, которые используются и до сих пор:

1. Машина должна работать не в десятичной системе исчисления (как механические арифмометры), а в двоичной (бинарной). Это означает, что программа и данные должны быть записаны в коде двоичной системы, где каждое число или символ представляется определенной комбинацией нулей и единиц.

2. Программа, которая управляет последовательностью выполнения операций, должна храниться в памяти машины. Там же должны храниться исходные данные и промежуточные результаты.

3. Чтобы достаточно быстро можно было считать, память компьютера следует организовать по иерархическому принципу, т. е. она должна состоять по крайней мере из двух частей: быстрой, но небольшой по емкости (оперативной) и большой (и поэтому медленной) внешней.

Согласно разработанному Дж. фон Нейманом и потом реализованному проекту ЭВМ должна состоять из следующих основных блоков:

устройства ввода-вывода, с помощью которого программа и данные записываются в память, а результаты выдаются оператору;

памяти, в которой хранятся данные и программы;

арифметического устройства, которое выполняет операции над данными;

устройства управления, которое управляет последовательностью операций в соответствии с выполняемой программой.

Так строят и современные компьютеры. Каждый из них имеет устройства ввода-вывода, память, арифметическое устройство и устройство управления (подробнее о них расскажем чуть позже). Эту схему компьютера, а ее чаще называют *архитектурой*, обычно связывают с именем ее автора Джона фон Неймана и назы-

вают *фоннеймановской.* Компьютер, отклоняющийся от этой схемы, называют нефоннеймановским.

В последнее время чаще всего нарушают второй принцип, требующий, чтобы процесс вычислений управлялся программой. В нефоннеймановской архитектуре управление вычислительным процессом происходит не программой, а получаемыми результатами — данными (об этом мы расскажем подробнее в гл. 6).

АЛГОРИТМ, ПРОГРАММА И ДАННЫЕ

С понятиями алгоритма, программы и данных нам предстоит встречаться многократно. Рассмотрим эти понятия подробнее.

Слово «алгоритм» появилось в средние века, когда европейцы впервые познакомились с работами великого арабского математика аль-Хорезми (783—855). Эти работы произвели на них столь глубокое впечатление, что появилось слово «алгоритм», которое происходит от имени (точнее, фамилии) этого ученого. Первоначально алгоритм не означал ничего более как нумерацию по арабской системе исчисления, с которой европейцы до тех пор не были знакомы. (Вспомним, каким открытием для европейцев явилась арабская система счисления, используемая нами поныне, именно она дала толчок в развитии техники вычислений.) Позже под алгоритмом стали понимать строго упорядоченное правило для превращения исходных данных в результат (например, известный всем алгоритм умножения многозначных чисел столбиком).

Итак, *алгоритм* — это точное правило, инструкция, указание, как нужно действовать, чтобы получить результат. Чтобы реализовать алгоритм в компьютере, нужно составить программу выполнения этого алгоритма и ввести ее в память компьютера.

Сам по себе (без программы) компьютер не способен решить ни одной задачи. Чтобы компьютер работал, прежде всего его нужно проинструктировать, какие операции и в какой последовательности нужно выполнять, т. е. составить программу его работы. Во-вторых, в память компьютера нужно ввести данные, с которыми ему предстоит работать. Таким образом, вводимые данные являются объектами, с которыми компьютер работает так, как указывает программа, т. е. в какой последовательности и что нужно сделать с этими объектами. Если компьютер решает вычислительную задачу, то, грубо говоря, на вопрос «что считать?» отвечают данные, а на вопрос «как считать?» — программа.

С подобными «программами» мы часто встречаемся в повседневной жизни. Например, кулинарная книга сплошь состоит из «программ», называемых рецептами. Рецепт для приготовления пражского салата можно записать как следующую программу. Данные: 150 г жареной телятины, 150 г жареной свинины, 150 г соленых огурцов, 150 г лука, 100 г яблок, майонез, лимонный сок или уксус. Программа: Все твердые компоненты нарезать, залить лимонным соком или уксусом и смешать с майонезом. Другую «про-

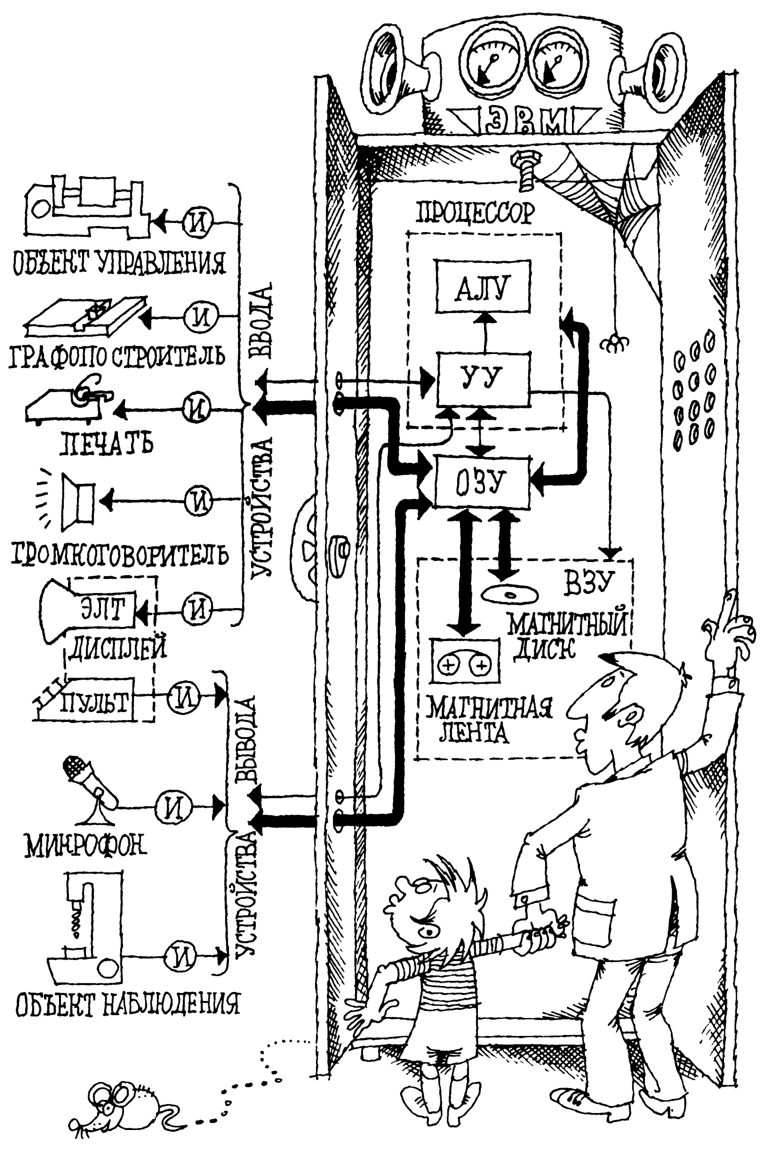

ЭВМ
ПРОЦЕССОР
АЛУ
УУ
ОЗУ
ВЗУ
МАГНИТНЫЙ ДИСК
МАГНИТНАЯ ЛЕНТА
ОБЪЕКТ УПРАВЛЕНИЯ
ГРАФОПОСТРОИТЕЛЬ
ПЕЧАТЬ
ГРОМКОГОВОРИТЕЛЬ
ЭЛТ
ДИСПЛЕЙ
ПУЛЬТ
МИКРОФОН
ОБЪЕКТ НАБЛЮДЕНИЯ
УСТРОЙСТВА ВВОДА
УСТРОЙСТВА ВЫВОДА
И

Рис. 1

грамму» дает мастер-слесарь своему ученику: «Возьми эту штуковину и приложи к той, что побольше. Затем скрепи их обе тем болтом». Здесь «данные» — это две детали и болт, а «программа» указывает порядок их соединения. И еще один пример. Мы просим своего близкого сходить за хлебом, кефиром и лекарством, даем только «исходные данные», а программу — последовательность, в какой посетить гастроном и аптеку, он составляет сам.

Таким образом, компьютерная *программа* состоит из набора инструкций (или команд, как принято называть в программировании), определяющих процесс переработки данных.

Следует отметить, что программы для компьютера нужно составлять намного подробнее, чем «программы» для человека. Составляя «программу» для человека, мы всегда рассчитываем, что ее исполнитель сам додумает то, что недосказано в программе, или переспросит, если что-то будет не ясно. Составляя же программу для компьютера, следует учесть все до последней мелочи. Поэтому в программах всегда много запасных вариантов, указывающих, что нужно делать в различных ситуациях. Подобные ветвления программы мы часто используем повседневно, например «Если нет кефира, купи простоквашу». Закодированное выражение такого рода в программе называют *логическим условием* или *условным переходом* и используют для того, чтобы перейти на ту или иную ветвь программы в зависимости от конкретных значений полученных данных. Так же, как факт отсутствия кефира в магазине вызывает программу покупки простокваши.

Читателю уже ясно, что все, на что способен компьютер, зависит от того, какие программы и данные находятся в его памяти, какую программу он выполняет в настоящий момент и какими данными оперирует. Поэтому точнее было бы говорить не о диалоге человека с компьютером, а о диалоге человека с программой, находящейся в этом компьютере.

А теперь заглянем внутрь компьютера. Мы уже говорили об атомах компьютера — логических элементах И, ИЛИ, НЕ. Самый маленький компьютер содержит 20—30 тыс. таких логических элементов, они образуют узлы, из которых и состоит компьютер.

Рассмотрим простейшую ЭВМ как автомат, имеющий лишь четыре узла (рис. 1): процессор, состоящий из арифметическо-логического устройства (АЛУ) и управляющего устройства (УУ), запоминающее устройство (ЗУ) и устройство ввода-вывода информации. Только выяснив их функции, можно понять, как работает ЭВМ и как функционирует машинный «разум».

АЛУ — ЭЛЕКТРОННЫЙ МОЗГ КОМПЬЮТЕРА

Центральной частью любого компьютера всегда было и остается АЛУ — арифметическо-логическое устройство, которое осуществляет основную долю обработки информации.

На вход АЛУ подается информация в виде кода операции —

оператора, который указывает, что должно делать АЛУ, и *операндов* — чисел или слов, с которыми этот оператор должен работать. На выходе АЛУ появляется результат этой операции. АЛУ работает как карманный калькулятор, который немедленно выдает результат, если в него ввести с помощью клавиш числа (операнды) и код операции (для каждой операции в микрокалькуляторе есть своя клавиша, помеченная одним из значков +, ×, и т. д.). Образно говоря, АЛУ является карманным калькулятором компьютера, но с большим числом «клавиш» для введения операндов и еще большим числом команд. Так, ЕС ЭВМ имеет АЛУ, выполняющую до 200 операций.

Если бы компьютер кроме АЛУ ничего не имел, то его уже можно было бы использовать, но только как мощный калькулятор с ручным управлением. «Нажимает» же «клавиши» АЛУ в компьютере управляющее устройство, действующее на основе информации, поступающей из запоминающего устройства, где хранятся все сведения, необходимые для его работы. Поэтому следующим по важности элементом компьютера является память.

ПАМЯТЬ КОМПЬЮТЕРА

Все знают, как велика роль памяти для организации целенаправленного поведения. Но если живое существо может некоторое время существовать без памяти (очень плохо, но может), то компьютер без памяти уже не компьютер — он не в состоянии сделать ничего (целесообразного, разумеется).

Мы уже говорили, что, прежде чем работать с компьютером, в его память необходимо ввести программу и исходные данные. Именно для хранения всей этой информации компьютеру необходима память. Функции такой памяти и выполняет запоминающее устройство (ЗУ). Имеется много типов ЗУ.

Запоминающее устройство адресного типа представляет собой набор перенумерованных ячеек памяти, где располагаются коды операторов программы и исходные данные. Каждая ячейка может хранить одно машинное слово. Если на вход ЗУ подать номер ячейки — ее адрес, то на выходе ЗУ появится двоичное число, кодирующее десятичное число, оператор, слово, которое является содержимым этой ячейки, записанным ранее. Это режим чтения информации в ЗУ. Так компьютер извлекает из своей памяти то, что в нее было заложено. В режиме записи на вход ЗУ подаются адрес ячейки и запоминаемое число (слово). В результате в эту ячейку будет записано именно это число. Для ЗУ все равно, что запоминать: числа, операторы или слова. Для него все это лишь набор двоичных символов.

Важной разновидностью ЗУ является *оперативное запоминающее устройство* (ОЗУ), в котором процессы записи и считывания происходят очень быстро. Это основная память компьютера.

Именно здесь хранятся программа и данные, необходимые для немедленного решения каждой конкретной задачи. Реализуется ОЗУ по-разному. Еще распространены магнитные ОЗУ. Они состоят из большого числа миниатюрных ферритовых колец, каждое из которых можно намагничивать в одном или другом направлении и этим запоминать один бит информации. Сколько колец в таком ОЗУ, столько бит информации оно может запомнить. Но магнитные ОЗУ слишком громоздки, и им на смену пришли более быстрые и компактные, состоящие из большого числа триггеров (триггер — специальная схема, способная запомнить один бит информации). При отключении электропитания электронное ОЗУ «забудет» всю информацию, что не происходит с магнитным ОЗУ. Однако это не смущает разработчиков, и будущее, несомненно, за электронным ОЗУ.

Емкость ОЗУ измеряется битами (по числу колец или триггеров), байтами (в одном байте 8 бит), но чаще в килобайтах (Кбайт) — тысячах байт (точнее, $2^{10}=1024$ байт). Так, емкость ОЗУ компьютера ЕС-1033, в зависимости от комплектации, 256 или 512 Кбайт. Часто емкость ОЗУ измеряется числом машинных слов (напомним, что обработка информации в компьютере происходит такими словами). Длина слова кратна байту, т. е. равна 8, 16, 32, 64 бит. Например, емкость ОЗУ компьютера СМ-4 в двухбайтовых словах —128 Кслов (килослов). Очень большая емкость памяти измеряется в мегабайтах (Мбайт) — в миллионах байт (точнее, $2^{20}=1\,048\,576$ байт). Например, емкость ОЗУ компьютера ЕС-1066, самого производительного ЕС ЭВМ, 16 Мбайт. Аналогично использование мегаслов (Мслов) для измерения емкости ОЗУ.

В любом компьютере именно ОЗУ обеспечивает гибкость его работы и быстродействие. Это быстрая память компьютера. От того, как быстро можно извлечь информацию из ОЗУ и записать в ОЗУ, зависит производительность компьютера. Но это не самая быстрая память.

Существует еще *сверхоперативная память* для временного хранения промежуточных результатов, которые вскоре могут понадобиться. Она представляет собой набор электронных ячеек памяти — регистров. Именно поэтому такую память часто называют *регистровой*. Сам по себе регистр — это последовательность триггеров, на каждом из которых запоминается один бит информации. Изготовляются регистры из быстродействующих элементов, что обеспечивает сверхоперативность запоминания и извлечения информации. Регистровый способ хранения данных широко применяется в каждом компьютере. Это связано не столько с производительностью, сколько с необходимостью иметь промежуточный буфер хранения информации при передаче от одного узла компьютера к другому. Так, передача информации из ОЗУ в АЛУ осуществляется обычно через два регистра, один из которых рас-

положен на выходе ОЗУ, а другой на входе АЛУ. И совместить их нельзя, так как информация из ОЗУ может быть направлена не только в АЛУ, но и на другие устройства. Аналогично АЛУ, получив результат, выдает его на выходной регистр и начинает работать с другой командой. А из выходного регистра АЛУ результат направляется туда, куда требует программа. Очевидно, что и в этом случае от регистра требуется высокое быстродействие, которое определяет скорость передачи данных в компьютере и в конечном счете сказывается на его производительности.

Другим видом запоминающего устройства в компьютере является *постоянное запоминающее устройство* (ПЗУ), которое используется для хранения программ и данных, необходимых постоянно для работы компьютера. Например, при вычислениях очень часто встречаются тригонометрические функции (синус, косинус и т. д.) Программы вычисления этих функций и хранятся в ПЗУ. Их составляют заранее и постоянно хранят. Аналогично поступают и с программами решения задач, с которыми приходится часто иметь дело, например программы для преобразования текста из одной формы в другую при редактировании. Тем и отличается ПЗУ, что допускает только считывание записанной информации.

И наконец, *программируемое постоянное запоминающее устройство* (ППЗУ). Оно используется для записи новых программ и данных, которые появляются и будут использоваться далее постоянно (так обычно бывает, когда с компьютером начинает работать новый пользователь со своими новыми запросами и потребностями). Этот вид памяти занимает промежуточное место между ОЗУ и ПЗУ.

Кроме этого, для хранения больших массивов информации и доступа к ним компьютер имеет еще внешние ЗУ (ВЗУ), обращение к которым занимает значительно больше времени: от 10^{-3} с до нескольких секунд или даже минут. (Заметим, что обращение к ОЗУ, ПЗУ и ППЗУ занимает 10^{-6}—10^{-8} с). Реализуется ВЗУ обычно на магнитных дисках или лентах, на которых можно записать в 10—1000 раз больше информации, чем в ОЗУ, но доступ к этой информации требует времени: нужный ее фрагмент сначала следует считать в ОЗУ и уж потом пользоваться. Очень удобны гибкие магнитные диски (дискеты или дискетки), очень похожие на маленькую грампластинку (диаметром около 20, 12 и 8 см). Они легко заменяются в компьютере (почти так же, как в проигрывателе). На каждый диск можно записать очень много информации, например при не очень плотной записи текст толстого журнала «Новый мир», а при плотной записи — текст его шести номеров.

В самое последнее время появились оптические диски, на которых информация записывается лазерным лучом. Такой способ позволяет записать чрезвычайно большие объемы информации (значительно больше, чем на магнитный диск), измеряемые

в гигабайтах — в миллиардах байт (1 Гбайт$=2^{30}$ байт). Это целая библиотека! Очень ценно, что записанную на оптический диск информацию нельзя случайно стереть или исказить под влиянием помех, так как оптическую запись нельзя нарушить электромагнитной или иной помехой, способной обесценить содержимое магнитной памяти.

И, наконец, самое медленное из ВЗУ (и самое дешевое) реализуется на магнитной ленте обычно в виде компакт-кассет, тех самых, которые широко используются для домашних магнитофонов. Поэтому очень часто для этой цели применяют стандартные бытовые магнитофоны. Перед тем как использовать информацию, хранящуюся на магнитной ленте, ее также нужно сначала перевести в ОЗУ.

Каждый вид памяти имеет свои достоинства и недостатки: ОЗУ представляет память быструю, но дорогую и поэтому малую по емкости, ВЗУ — медленную, но зато «дешевую». Получается так, что чем больше емкость памяти, тем труднее доступ к ней, тем дольше приходится ждать, пока ее содержимое попадет в ОЗУ и станет доступным для работы компьютера. Но самую большую емкость памяти имеют магнитные ленты, хранящиеся обычно на полке в «магнитотеке». Однако сначала их нужно там найти, поставить на магнитофон, перевести информацию на более быструю память — магнитный диск и, наконец, с него в ОЗУ. Это долго, но гарантирует возможность оперировать огромными объемами информации.

Таким образом, современный компьютер оснащен большим числом ЗУ различного назначения, быстродействия и емкости. Если в основу положить быстродействие ЗУ, то получится следующий ряд: регистровое ЗУ, ОЗУ, ПЗУ, ППЗУ, магнитные (или оптические) диски и, наконец, магнитные ленты.

Работа компьютера, по сути дела, сводится к организации взаимодействия АЛУ (знающему, «как делать») и ЗУ (знающему «что делать»). Однако ЗУ и АЛУ сами по себе взаимодействовать не могут. Это взаимодействие обеспечивает ...

УПРАВЛЯЮЩЕЕ УСТРОЙСТВО КОМПЬЮТЕРА

Управляющее устройство (УУ) взаимодействует со всеми узлами компьютера и организует передачу информации от одного узла к другому для выполнения программы, записанной в ОЗУ. Забот у него достаточно. Действительно, каждый узел компьютера специализирован на выполнении определенных функций, которые реализуются лишь при получении необходимой информации. Так, для работы АЛУ нужно передать на исполнение *команду* — код операции и необходимые данные (операнды), с которыми должна быть произведена эта операция. Запоминающее устройство выдает информацию и запоминает ее при соответствующем обращении к нему. Устройства ввода-вывода начина-

ют функционировать и передавать необходимую информацию в указанный узел (в ОЗУ, ВЗУ и т. д.) лишь при соответствующих командах. Все эти операции осуществляет управляющее устройство. При работе оно, прежде всего, пользуется «указаниями» выполняемой программы, расположенной в ОЗУ,— извлекает операторы и операнды из ОЗУ и направляет их в АЛУ, а результат в соответствии с указанием программы либо направляет в ОЗУ, либо реагирует на этот результат определенным образом. Например, включает устройство ввода для «подкачки» необходимой информации из ВЗУ, или выдает полученную информацию на экран пользователю, или останавливает работу компьютера (например, при делении на нуль или при окончании счета).

Но основной «заботой» УУ является все же загрузка АЛУ, ведь именно бесперебойной работой АЛУ определяется производительность компьютера. Объединение УУ с АЛУ называют *процессором*. Конечно, в процессоре есть и другие элементы, но основными, безусловно, являются АЛУ и УУ.

На процессор и возлагаются функция выполнения процесса обработки информации и программное управление. Следовательно, процессор и ОЗУ обеспечивают основную функцю компьютера — обработку информации. Но для взаимодействия с окружающим миром ему нужны «органы чувств». Без них он «глух и нем». Функцию органов чувств компьютера выполняют устройства ввода-вывода информации. Рассмотрим их.

УСТРОЙСТВА ВВОДА

С помощью этих устройств в компьютер (а точнее, в его память — ОЗУ и ВЗУ) засылается информация. Способов организации такого ввода много. Но прежде всего заметим, что соединение любого устройства ввода (и вывода) с УУ производится через специальное устройство — *интерфейс* (от англ. inter — между и face — лицо). Интерфейс —«межличностное» устройство, играет роль своеобразного «переводчика» с «языка» одного устройства на «язык» другого и позволяет сопрягать различные устройства для обмена информацией между ними.

Интерфейс в широком смысле — правила взаимодействия различных устройств. Эти правила можно реализовать программно и тогда будем иметь программу интерфейса. Правила интерфейса аппаратурно реализуются интерфейсным устройством. Например аналого-цифровой преобразователь (АЦП), преобразующий электрический ток в его цифровое представление в двоичном коде, является типичным интерфейсом между электрическим процессом и компьютером, который перерабатывает этот процесс.

А теперь вернемся к устройствам ввода (см. рис. 1).

Пульт является наиболее распространенным и очень удобным видом устройства ввода информации в компьютер. Он похож на

обычную пишущую машинку, но с 50—80 клавишами. Кроме букв и цифр, на клавиатуре пульта всякого рода знаки, нужные для программирования, например, =, >, <, ×, #, ¤ и т. д., а также знаки →, ←, ↑, ↓, с помощью которых можно указанным образом изменять положение курсора. *Курсор* — это яркое пятнышко на экране компьютера, указывающее, где появится следующий знак при нажатии клавиши. Именно курсор позволяет пользователю располагать знаки в нужной ему точке экрана. Сигналы от каждой клавиши преобразуются в коды данных (цифр, букв, значков) и операторов с помощью интерфейса пульта. Этот интерфейс образуется схемами преобразования сигнала от каждой клавиши в соответствующие коды данных или операторов, которые и поступают в ОЗУ.

В последнее время ведутся интенсивные работы в области создания устройств ввода голосом через микрофон. Это акустический ввод данных, которые наговариваются пользователем компьютера. Но для этого необходимо иметь преобразователь электрических колебаний, образованных микрофоном при произнесении слов, в числа и операторы программы. Так, слово «один» должно быть преобразовано в цифру 1, а произнося слово «стоп», мы вызываем оператор прерывания работы программы. Стоит ли говорить, что интерфейс такого ввода должен решать очень сложную задачу понимания и перевода слов с человеческого языка на машинный, да еще с учетом особенностей произнесения тем или иным человеком. Она решается или с помощью специального компьютера, или на том же компьютере, с которым работает говорящий. В последнем случае предварительный интерфейс преобразует колебания тока микрофона в последовательность цифр, кодирующих эти колебания, которые попадают в ОЗУ. Эти цифры затем обрабатываются процессором по специальной программе, результатом которой и является побуквенная запись в ОЗУ произнесенного слова или фразы.

И, наконец, если компьютер используется для контроля или управления каким-то объектом, в него нужно вводить показания приборов и датчиков, определяющих состояние наблюдаемого объекта. Здесь в качестве интерфейса используют устройства связи компьютера с объектом.

Следует заметить, что легкосъемные ВЗУ, такие, как компакт-кассеты и магнитные диски, тоже могут служить устройствами ввода информации в компьютер, но не любой информации, а только той, которая ранее была получена с помощью компьютера. Например, созданной вами программой заинтересовался кто-то другой. Ему для работы с ней вовсе не нужно садиться за ваш компьютер, где в ОЗУ находится интересующая его программа. Достаточно «сбросить» ее в ВЗУ на компакт-кассету или гибкий диск, забрать это ВЗУ, вставить его в свой компьютер и переслать информацию с ВЗУ в ОЗУ. Вот и весь ввод! Именно

такой способ используется сейчас для введения в компьютер новых программ (чаще всего игровых), разработанных специально для массового применения потребителями в виде гибких магнитных дисков или, реже, компакт-кассет.

А теперь рассмотрим устройства вывода. Если без ввода компьютер слеп и глух, то без вывода он нем, а поэтому бесполезен.

УСТРОЙСТВА ВЫВОДА

Одним из наиболее совершенных средств вывода является *электронно-лучевая трубка* (ЭЛТ). Она практически не отличается от телевизионной. На экран ЭЛТ выдаются тексты (состоящие из слов и цифр), необходимые для работы пользователя при вводе исходной информации (для контроля), при отладке процесса решения его задачи (при выводе промежуточных результатов), а также при выводе окончательных результатов обработки.

Для фиксации информации издавна в ЭВМ используются печатающие устройства — *принтеры*, которые позволяют выводить информацию (цифровую, текстовую, графическую) на бумагу. Простейшим принтером является электрическая пишущая машинка.

В системах автоматизированного проектирования (САПР) конечным продуктом являются чертежи спроектированного компьютером изделия, агрегата, детали... . Здесь устройство вывода должно выдавать конкретные чертежи, что и делает *графопостроитель*. Это автоматическая чертежная доска, управляемая компьютером. Интерфейсом к ней служат очень сложные преобразователи цифр, кодирующих чертеж, в команды управления двигателями, передвигающими рапидограф, — чертежный инструмент, напоминающий фломастер.

Для управления объектом с помощью компьютера (например, в гибких автоматизированных производствах) необходимо иметь возможность автоматически обрабатывать его команды. Это выполняют исполнительные механизмы на объекте (например, сервоприводы), они быстро и точно переводят управляемый орган в положение, которое определено компьютером (например, передвигают резец в заданное положение, включают двигатель, перемещают деталь в определенное место). В этом случае интерфейс преобразует найденное компьютером число в величину, определяющую положение управляемого органа (резца, детали и т. д.) и выдает команды соответствующим исполнительным механизмам.

В последнее время появилось еще одно устройство вывода. Новым назвать его нельзя — это обычный громкоговоритель, на который возлагается функция речевого общения с пользователем. Интерфейс этого устройства должен преобразовывать

числа, слова и фразы, образованные в памяти компьютера, в электрические колебания, соответствующие человеческому голосу, читающему эти числа, слова, фразы. Такой преобразователь текста в речь называют *синтезатором речи* (его функция обратна речевому вводу, упомянутому ранее). Это сложное устройство обычно реализуется с помощью специального компьютера, который и образует нужный интерфейс.

Одним из самых распространенных в настоящее время устройств ввода-вывода является *дисплей* (англ. *display* — показ). Строго говоря, дисплей — средство отображения и представления информации. Но сейчас понятие дисплея расширилось и на ввод информации с помощью клавиатуры. Дисплей является сочетанием алфавитно-цифровой клавиатуры пульта (это ввод информации) и экрана ЭЛТ или экрана на жидких кристаллах (это вывод). Пульт и ЭЛТ — самые крупные детали современных компьютеров, которые делать меньше просто нецелесообразно, так как пользователю трудно будет работать. А так как остальные узлы ЭВМ (процессор и ЗУ) изготовляются в виде микросхем и поэтому имеют очень малые габаритные размеры, то иногда весь компьютер размещается в коробке самого дисплея.

В зависимости от области применения, класса задач, потребностей, квалификации и вкусов пользователей и других факторов изготовляются самые разнообразные компьютеры — от карманных до больших шкафов-стоек. Но каждый из них всегда имеет указанные узлы — процессор, ЗУ и устройства ввода-вывода.

Мегрэ поморщился:

— Ну и «собеседник»! Одни железяки да пластмассовые кнопочки!— и он с опаской посмотрел на дисплей компьютера.— Красив, слов нет! Но что толку в этом?

— Поль,— раздраженно позвал Мегрэ,— ну-ка покажи возможности этого «собеседника»! Пусть он определит, не числится ли в досье нашего комиссариата погибшая. Только не надо «колдовать»— объясняй все толком!

— Слушаюсь, шеф,— улыбнулся Поль и вытащил из шкафа тонкий черный бумажный квадратный пакет с отверстием посредине.

— Это дискетка, на ней записаны данные на всех «клиентов» нашего комиссариата. Я ее вставляю в дисковод компьютера.— И Поль начал вводить пакет в узкую щель на передней панели компьютера.

— Постой!— остановил его Мегрэ.— Ты же забыл вынуть дискетку из конверта.

— А она не вынимается. Видите, это широкая поперечная прорезь — окно в конверте, через которое виден сам диск. Через него и считывается информация. А сама дискетка вращается дисководом внутри конверта со скоростью 300 оборотов в минуту — это в 10 раз быстрее, чем пластинка в электрофоне. Считывающая головка движется вдоль прорези и может останавливаться на каждой из 35 дорожек диска. Нам понадобится какая-то определенная часть содержимого этого диска? Или все?

— Трудно сейчас сказать. А что, это важно?

— Да. Ведь для работы с этими данными их нужно перевести с диска в оперативную память компьютера — его ОЗУ, а эта память ограничена.

— Выходит, что работать с массивом информации объемом больше, чем емкость ОЗУ, нельзя?— удивился Мегрэ.

— Можно, но немного дольше. Ведь при каждом обращении к информации на диске нужно подвести головку к соответствующей дорожке и считать ее. А на это затрачивается время.

— Ладно. Выведи все, что известно о нашей несчастной, если она там есть, разумеется.

Поль набрал на клавиатуре имя и фамилию погибшей. На экране дисплея через мгновение появилось досье. Мегрэ внимательно прочел все. При этом пришлось несколько раз «продвигать» текст досье по экрану, так как оно занимало несколько страниц, каждая из которых соответствовала размеру экрана.

Девушка однажды проходила по мелкому делу о продаже наркотиков, поэтому она и попала в досье.

— Поль! Давай-ка полистаем досье на ее соучастников. Нет, их слишком много. А нельзя ли узнать, кто из них сейчас на воле?

— Пожалуйста, месье!— И на экране появились три фамилии.

— А теперь узнайте, кто из них проходил по делу убийства президента страховой компании «Космос» в прошлом году.

Поль быстро набрал на экране год и слова «Космос», «убийство президента». Быстро защелкал привод магнитной головки, считывающей информацию с гибкого диска, и скоро на экране появилось «Жан Волье — Весельчак».

— Вот он-то мне и нужен. Скорее всего, именно он главный в этой истории, а не этот прыщавый юнец, который лишь выполнил его приказ. Действуйте, Поль.

— Слушаю, шеф,— по-военному отрубил Поль.

—Что ж, подумал Мегрэ,— компьютерное досье, действительно, удобная штука. Представляю, сколько времени пришлось бы возиться, чтобы найти этого Жана «старым» способом. Нужно освоить эту штуковину и иметь ее у себя на столе.

— Поль!— окликнул Мегрэ.— А нельзя ли заказать такой компьютер для нашей группы, для коллективного пользования?

— Нет. А вот для персонального — можно. Это будет совсем небольшой компьютер. Не больше среднего телевизора. Его так и называют «персональный компьютер».

Что это такое, вы узнаете дальше.

2. МОЙ, ТОЛЬКО МОЙ
(Персональный компьютер)

КОМПЬЮТЕРЫ БЫВАЮТ РАЗНЫЕ

Возникнув 40 лет назад, они довольно долго были машинами для немногих избранных. Об их возможностях слагались легенды, и широкие круги научных работников, инженеров и просто заинтересованных лиц даже мечтать не могли о том, чтобы

воспользоваться компьютером для решения своих насущных задач. Препятствий этому было много: малое число машин, их загруженность важными техническими и научными задачами, малая производительность и небольшая память компьютеров того времени. Но, пожалуй, самым главным препятствием была дороговизна первых ЭВМ, что и определяло высокую стоимость машинного времени. Чтобы приблизить компьютер к широкому кругу пользователей, нужно, прежде всего, снижать его стоимость. Так начался и до сих пор продолжается грандиозный процесс усовершенствования технологии изготовления компьютеров, направленный на улучшение его характеристик и, прежде всего, снижение стоимости. Ни одна отрасль промышленности не развивалась столь быстро, и нигде характеристики продукции не улучшались столь интенсивно, как в компьютерной технике! Этот процесс можно охарактеризовать цифрой 10 — за 10 лет в 10 раз улучшились характеристики компьютеров: в 10 раз уменьшились их объем и стоимость, в 10 раз увеличились производительность и емкость памяти и т. д. Такого темпа не было нигде и никогда!

Вычислительная техника развивалась по нескольким направлениям. Каждое из них связано с решением важных задач и удовлетворением насущных потребностей человечества. Так, необходимость решения задач большой сложности (например, расчет и коррекция траектории космической ракеты или оперативной обработки больших потоков информации) привела к созданию суперЭВМ с производительностью в сотни миллионов операций в секунду (правильней их назвать вычислительными системами).

Персональные компьютеры возникли в результате усилий по приближению ЭВМ к широким кругам пользователей (точнее, потребителей, если точно переводить английское слово user, породившее наше «пользователь»). Именно так называют всякого, кто обращается к компьютеру для решения своих проблем, профессиональных и бытовых.

Короткая, но бурная история персональных компьютеров началась в 70-е годы. Появился новый тип ЭВМ — мини-ЭВМ. Они возникли на базе интегральных микросхем, в каждой из которых тысячи логических элементов. Упрощение проявилось в «укорачивании» разрядности мини-ЭВМ до 8—16 бит (по сравнению с 32—64 бит «больших» ЭВМ) и в уменьшении числа выполняемых команд. Однако такие упрощения не привели к снижению вычислительных возможностей. Так, требуемая точность мини-ЭВМ обеспечивается путем использования для представления чисел нескольких машинных слов (удвоение, утроение и т. д. точности расчетов), а все недостающие арифметические операции (например, умножение и деление) выполняются соответствующими программами, что несколько снижает производительность компьютера, но упрощает его схему (точнее, схему АЛУ).

Примерами таких мини-ЭВМ в нашей стране являются ма-

шины серии СМ, которые сегодня пользуются заслуженной популярностью. С появлением мини-ЭВМ число машин начало быстро расти. Но и этого было недостаточно. Решающим стало создание микропроцессоров (1971 г.), которые открыли возможность значительно упростить и удешевить компьютер. Так появились микроЭВМ (формально — это компьютер, построенный на базе микропроцессора, хотя микропроцессоры сейчас используют широко и в мини- и в макси-ЭВМ). МикроЭВМ стала следующим шагом, приближающим компьютер к массовому пользователю. Именно эти машины стали основой для действительно массовой ЭВМ — персонального компьютера (ПК). Стремление к массовости и привело к появлению минимального компьютерного комплекта — самого простого персонального компьютера, бытового.

МИНИМАЛЬНЫЙ КОМПЛЕКТ

Почти в каждом доме есть магнитофон и телевизор. Их очень естественно использовать в качестве устройства ввода-вывода информации: магнитофон — ввод, телевизор — вывод. Так возникает простая, но плодотворная идея минимального компьютерного комплекта: пульт вместе с вмонтированным в него процессором и ОЗУ и телевизор с магнитофоном. Именно этот компьютерный комплект обычно называют *бытовым компьютером* или просто БК. Обычно он используется для всякого рода телевизионнных игр. Сама игра·и звуковое сопровождение к ней записаны на компакт-кассету, ее нужно вставить в магнитофон, ввести в память (ОЗУ) и играть.

Но такой мини-компьютерный комплект может быть использован и для решения серьезных задач обработки информации, таких, как вычисления, работа с массивами информации (например, ведение библиографии по своей специальности или работа со своей записной книжкой, содержание которой записано на компакт-кассету), обработка текстов, например при их редактировании, и многое другое. Указанные неигровые функции БК являются лишь дополнением к основной игровой и поэтому развиты слабо. Так, объемы редактируемого текста или записной книжки не могут быть большими: для этого не хватит памяти — ее достаточно лишь для игр.

Всем хорош мини-комплект, но... не очень удобен — его нужно разбирать, если захочется посмотреть телевизор или послушать магнитофон, да и времени трубуется слишком много. Так, для отыскания нужной информации, записанной на компакт-кассете, требуется 30 мин, т. е. то самое время, за которое прокручивается одна сторона кассеты на бытовом магнитофоне. Это еще не персональный компьютер, а лишь попытка обойтись «подсобными средствами». Это и многие другие обстоятельства привели к появлению нового типа компьютера. Персональный компьютер —

первый массовый компьютер для персонального применения. Ранее все компьютеры разрабатывались для коллективного пользования: на одну и ту же машину могли и должны были выходить по расписанию со своими задачами разные пользователи. Но такое коллективное разделение машинного времени можно сравнить с коммунальной квартирой, жить в которой никто не хочет, даже с самыми милыми соседями. И дело тут вовсе не в склочности их характера. Просто коммунальность квартиры требует по своему определению разделения общих ресурсов (кухни, коридора, ванной, туалета и т. д.), что, естественно, всегда неудобно. Персональный компьютер подобен отдельной квартире — все ресурсы находятся в вашем безраздельном пользовании.

ОБЛИК ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

Персональный компьютер представляет собой настольную микроЭВМ, внешне очень похожую на обычный дисплей — ТВ-экран и клавишный пульт. Он обязательно имеет внешнюю память (ВЗУ) на гибком диске, которая используется в качестве дополнительного устройства ввода-вывода стандартных программ и для хранения полученных и промежуточных результатов, массивов данных и программ.

Дополнительно к ПК придаются модули расширения. Это печатающее устройство, графопостроитель, устройство для дополнительных ВЗУ в виде гибких дисков, накопитель на магнитной ленте (магнитофон с компакт-кассетами), средства связи с другими ПК по телефонным и другим линиям связи и т. д. Но и без этих дополнений ПК оказался великолепным инструментом, с помощью которого можно повысить эффективность интеллектуальной деятельности человека.

Внутри ПК устроен еще проще: микропроцессоры, ОЗУ и микросхемы интерфейсов. Как правило, ПК снабжен накопителем на жестком несъемном диске типа винчестер (любопытно происхождение этого названия: первоначально этот диск имел обозначение «30/30», т. е. такое же, как у ружья марки «Винчестер»). Жесткий винчестерский диск, помещенный в герметический корпус, вращается со скоростью в 10 раз большей, чем гибкий диск, что обеспечивает быстрый доступ к записанной на нем информации. Емкость винчестерского диска очень велика — до 10 Мбайт, т. е. в тысячи раз больше, чем гибкого диска.

Появился ПК в 1973 г. в виде довольно экзотической и дорогостоящей игрушки, а с 1976 г. началось его триумфальное шествие. Потребности в ПК и его производство возрастали невиданными темпами. Так, в 1983 г. в мире было продано 10 млн. ПК — цифра небывалая даже для вычислительной техники!

ЗАЧЕМ НУЖЕН ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

Различают бытовые (домашние) и профессиональные ПК. Бытовые используются в качестве домашнего информационного центра. Это, прежде всего, развлечения (игры), учебные курсы для детей и подростков, обучение иностранному языку, расчеты семейного бюджета и многое другое.

Профессиональные ПК сначала получили применение для обработки текстов, подготовки отчетов (т. е. таблиц, диаграмм, графиков и т. д.), автоматизации делопроизводства. Но постепенно они проникли в сферу индивидуальной обработки инженерной, медицинской и другой информации, преподавательской деятельности в школах и вузах. Применяются они и в практике лабораторных научных экспериментов и т. д.

Пользователи ПК делятся на программирующих и непрограммирующих. Последние не умеют программировать, но хотят пользоваться услугами компьютера. Очевидно, что вторых значительно больше, чем первых. Это учтено в программном обеспечении ПК. Именно на них он и рассчитан.

В чем же может помочь ПК? Чтобы разобраться в этом, рассмотрим его возможности.

Основной и единственной формой общения пользователя с ПК является диалог. С помощью клавиатуры пользователь вводит в ПК данные в виде текстов, они тут же воспроизводятся на экране дисплея. Диалог можно вести на разных языках: меню, естественный язык для неискушенных пользователей и алгоритмические языки высокого уровня (обычно это Бейсик) для искушенных (об этих языках см. гл. 5).

«МЫШЬ» В ПЕРСОНАЛЬНОМ КОМПЬЮТЕРЕ

Очень часто при работе в режиме меню в ПК используется «мышь»— манипулятор в виде небольшой обтекаемой коробочки, похожей на мышь, с кнопкой и «хвостом»— шнуром, соединяющим ее с компьютером. Это очень удобное средство введения информации для пользователей, которые не умеют работать даже с клавиатурой. «Мышь» можно легко катать по гладкому столу. При этом на экране дисплея движется курсор (мы о нем говорили в предыдущей главе) в виде изображения мыши (это, разумеется, не обязательно, но забавно). Положение курсора на экране, таким образом, можно легко изменять, помещать его в любую точку экрана.

Используют «мышь» в режиме меню следующим образом. Изображение «мыши» подводят к номеру выбранного пользователем варианта ответа в предложенном меню и нажимают кнопку на «спинке» мыши. Это и есть действие пользователя, эквивалентное нажатию клавиши с соответствующим номером на пульте.

«Мышь» может помочь не только в выборе ответа в меню. С ее помощью можно рисовать на экране дисплея кривые, изменять положение картинок на экране, перетаскивая их в нужное место, стирать изображения с экрана (и в памяти компьютера, разумеется) и т. д. Каждое из этих действий зависит от предварительного выбора в режиме того же меню.

В общем, «мышь»— очень удобный инструмент общения с компьютером в режиме диалога и без труда, сразу же осваивается любым пользователем.

ОБУЧЕНИЕ РАБОТЕ С СОБОЙ

Персональный компьютер может также обучать пользователя, как нужно работать с ним и его программами. Прежде чем сесть за дисплей неперсонального компьютера, нужно прочитать и выучить довольно пухлую инструкцию к нему. А за ПК можно садиться на любой стадии обученности программированию (в том числе и нулевой): он быстро оценит степень вашей осведомленности и обучит этому нехитрому делу настолько, насколько вам это необходимо для дальнейшей работы с ним. Это значит, что ПК снабжен мощными программами обучения пользователя всем правилам общения с ним и способам манипуляции с его агрегатами, такими, как дисковод магнитных дисков, принтер, позволяющий переводить на бумагу то, что изображено на дисплее (в том числе и картинки), и многому другому, что нужно знать для эффективного взаимодействия с ПК.

При затруднениях пользователю достаточно нажать клавишу со словом «помощь», и ПК на экране даст справку-подсказку и на примерах покажет, как можно или следует поступить в сложившейся ситуации. Например, вы достали дискетку с нужной вам программой, например захватывающей компьютерной игры, но не знаете, как ее запустить. Наберите слово «дискета» на дисплее и нажмите клавишу HELP (Помощь). Компьютер сразу поймет, что у вас трудности с дискеткой и предложит меню вопросов, чтобы выяснить, что именно у вас не получается.

ДИСПЛЕЙ — ЛИЦО ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

И лицо очень симпатичное! Прежде всего оно обычно цветное (хотя есть еще ПК с черно-белым экраном). Цветной экран позволяет очень эффективно подавать информацию пользователю, обращая его внимание на ту или иную часть экрана. Манипулируя цветом текста и фона, ПК создает красочные композиции из простого текста и тем самым воздействует на эмоциональное состояние пользователя, повышая тем самым производительность его работы.

Очень интересна особенность ПК —«многооконный экран».

Этот режим моделирует на экране обстановку рабочего стола, на котором расположены материалы, нужные для работы, например для составления какого-то документа. Вспомогательные материалы, если их много, могут частично перекрывать друг друга, образуя те самые «окна», сквозь которые можно их читать. Пользователь может сдвигать и расширять эти окна, «вытаскивать наверх» нужные документы, убирать их и т. д., т. е. делать на экране все то, что мы делаем на столе во время работы. Здесь же может находиться калькулятор (точнее, его изображение на экране дисплея), с которым легко манипулировать с помощью курсора — «мыши», т. е. «нажимать» любую клавишу калькулятора. При этом результат будет появляться на индикаторе калькулятора — изображении на экране ПК. Такой многооконный экран позволяет пользователю эффективно общаться с ПК в процессе решения тех его задач, в которых нужно «иметь под рукой» несколько источников информации.

— Все эти необыкновенные возможности совершенно меня не вдохновляют,— зевнув заметил Мегрэ.— Уж лучше я разложу материалы на своем столе. Это и проще, и привычнее. Да и для манипуляции с текстом на экране наверняка надо знать какие-то правила. Нет, это не для меня. И не называйте меня ретроградом. Мне противно менять свои привычки. Без большой необходимости.

— Вот вы и проговорились. Не такой уж вы ретроград, если понимаете эту необходимость. А что эта необходимость большая, легко доказать.

— Попытайтесь,— улыбнулся Мегрэ.

— Сколько времени у вас отнимает составление ежемесячного отчета?

— Обычно дня два,— вздохнул Мегрэ.— Но здесь ваш компьютер не поможет. Просто я не могу сразу написать отчет набело, всегда хочется что-то изменить. Вот и приходится не раз перепечатывать, за что и сердится наша машинистка Мадлен.

— Да ведь этого никто не может, без многократной коррекции не обойтись. Причем зависит это от многих факторов, как объективных, так и субъективных.

— Все это так,— нетерпеливо заметил Мегрэ.— Но какое отношение это имеет к дисплею?

— Самое прямое. Составление текста и его редактирование на экране гарантирует, что этот текст будет таким же и на бумаге.

— И какой выигрыш при этом?

— Вам интересен опыт писателя-фантаста Айзека Азимова?

— Конечно. Его работоспособность меня всегда поражала — он в год выпускает 10—12 книг. Говорят, в его кабинете четыре машинки, на каждой он пишет свою рукопись.

— Писал! Не волнуйтесь — он жив, но пишет уже на компьютере. Теперь на книгу у него уходит недели две.

— Теперь у него четыре компьютера?

— Нет, один. Но все рукописи хранятся в памяти компьютера. Чтобы перейти от одной к другой, он вызывает на экран файл нужного текста и на одном компьютере может писать не четыре книги, а больше.

— И не нужно вставлять бумагу, вкладывать копирку, следить за полями...,— задумиво сказал Мегрэ.

— Разумеется! Всю рутину возьмет на себя компьютер.

— А не думаете ли вы, Поль, что творчество дело более тонкое, и бывает, что и рутинные операции необходимы — в это время вы осмысливаете созданное.

— Ну это что-то из области психологии и не имеет отношения к компьютеру.

— И очень плохо, что не имеет,— грустно закончил Мегрэ.

ИГРА — ДЕЛО СЕРЬЕЗНОЕ

Нет, наша жизнь не игра, как это легкомысленно считал Германн из «Пиковой дамы». Но игры занимают в ней значительное место. Дело в том, что человеку свойственно, прежде чем принять какое-то решение, представить его последствия. А чтобы «представить», надо шаг за шагом проследить, как будут развиваться «события» во всех возможных вариантах. А это, как легко заметить, не что иное, как игра, в которой ходами являются принимаемые решения, а ответами — реакция среды на них. Именно поэтому часто говорят, что для принятия какого-либо ответственного решения нужно «разыграть» все его последствия и оценить тяжесть каждого из них. С этим и связан интерес людей к играм как к моделям реальной жизни, в которой приходится им действовать. И дети играют нисколько не больше, чем взрослые, просто игры у нас с ними разные.

Это и определяет всеобщий интерес к компьютерным играм.

Имеется огромное количество игр, рассчитанных на самые различные интересы самых разных пользователей. И это далеко не только забава.

В компьютерных играх пользователь овладевает прежде всего процессом общения с компьютером, узнает его возможности, учится программировать и использовать другие программы. Игровые возможности ПК позволяют не только играть в конкретные игры, но и реализовать принцип «работать, играя», значительно повышающий производительность пользователя ПК. Появилось такое новое понятие «поп-программа», которая, сочетая игру с решением конкретной задачи, стимулирует интерес пользователя к самостоятельному созданию программ, даже если он непрограммирующий.

Действительно, овладев одной, другой, ... , десятой игрой, активный пользователь наверняка захочет кое-что изменить в них, сделать игры интересней. Но для этого надо изменить правила игры, ввести новые и т. д. А это можно реализовать, лишь изменив программу. А для этого нужно уметь программировать — овладеть одним из языков программирования (об этих языках мы поговорим подробнее в гл. 5). Вот и получается: хочешь по-новому играть с компьютером — научись программировать.

Более того, игру можно придумать самому и запрограммировать ее с помощью того же ПК. Большинство компьютерных игр придуманы и запрограммированы самими пользователями, которые в результате становились квалифицированными программистами.

2 Зак 2074

Поучительна история возникновения одной из крупнейших фирм по производству ПК, имеющей странное название Apple (яблоко). Она возникла в гараже, где два молодых техника игровых электронных автоматов решили на их базе создать компьютер, который оказался одним из первых массовых ПК. В нем игровая компонента значительно превышала то, что допускали «солидные» фирмы. Это и определило успех «гаражного предприятия». Это не реклама американской мечты, а вполне реальная ситуация, когда потребности человечества в ПК не были угаданы мощными фирмами (такими, как IBM), которые лишь позже развернули массовое производство ПК.

И, наконец, ПК обычно снабжен возможностью звукового сопровождения процесса взаимодействия с пользователем. Это выстрелы и взрывы в играх погони, восторженные реплики при удачных ходах, музыка и т. д. Звуковые подсказки или успокаивающая мелодия при благополучной работе чрезвычайно благоприятно действуют на пользователя, снижают нервозность, неизбежную при общении неискушенного человека со сложной машиной, особенно в режиме обучения.

Все эти свойства ПК создают обстановку дружественности общения человека и компьютера. Это обстоятельство породило новое понятие дружественного компьютера, что целиком относится именно к персональному компьютеру.

ГИБКИЕ ДИСКИ — СМЕННАЯ ПАМЯТЬ КОМПЬЮТЕРА

Сам по себе персональный компьютер умеет немного. Разве что предоставляет возможность программирующему пользователю самому составить программу на одном из алгоритмических языков (обычно Бейсике — о нем в гл. 5). Почти все свои «умения» ПК приобретает довольно просто — за счет соответствующих программ, считываемых с гибких дисков. (Напомним, на диск можно записать большой объем информации, значительно больший, чем может вместить ОЗУ компьютера. Поэтому с диска в ОЗУ за один прием снимается лишь часть информации, не больше свободной емкости ОЗУ.)

Что же записывают на гибких дисках? Да все, что может понадобиться пользователю при его общении с ПК. Прежде всего, игры. Их создано очень много. С них-то и началось массовое увлечение ПК как универсальным инструментом проигрывания игротеки, записанной на гибких дисках. Но не «играми едиными» жив ПК. Его основная функция — помогать пользователю в решении нужных ему задач. Программы решения этих задач объединяются в так называемые *пакеты прикладных программ* (ППП) — пакеты программ решения конкретных прикладных задач. В мире более 25 тыс. ППП, из них половина написана специально для ПК. Приведем примеры наиболее распространенных.

Пакеты программ текстовой обработки

Они созданы для составления и редактирования текстов. Это одна из самых распространенных невычислительных функций компьютера. Именно поэтому программу текстовой обработки называют часто текстовым процессором (обратите внимание: процессор не только аппарат, но и средство переработки информации). Он предназначен для подготовки и обработки всех видов текстовой информации — писем, статей, описаний, инструкций и т. д. С помощью ППП текстовой обработки можно вводить, редактировать и компоновать любой текст. Подготавливаемый или уже готовый текст можно хранить на гибком диске, на котором можно разместить до 250 машинописных страниц, или выводить на печать с помощью принтера. При этом текст будет напечатан точно в таком же виде, в каком он был на экране дисплея. Это особенно удобно при составлении документов и статей.

Программа текстового процессора позволяет осуществлять всякого рода манипуляции с текстом, например заменить всюду в тексте одно слово на другое, вставить новое слово или фразу, поменять местами слова или предложения, переформатировать текст на стандартной бумаге, проставляя номера страниц, и т. д. Такого рода текстовые процессоры очень помогают всем, кому приходится много писать (журналистам, ученым, писателям, администраторам и т. д.), а так как таких пользователей очень много, выпускаются специальные ПК, ориентированные только на текстовую обработку.

Пакеты программ графики

Очень популярен ППП деловой и инженерной графики, позволяющий представлять данные в наглядной графической форме — в виде графиков и диаграмм. Дело в том, что существует сравнительно немного видов графического представления информации. Это линейный график (обычная зависимость какого-то показателя от одного фактора), столбцовый график, где размер столбца характеризует уровень какого-то важного показателя, круговой график (или секторная диаграмма), позволяющий наглядно показать чьи-то доли в виде секторов круга и т. д.

Пакет программ графики позволяет строить и комбинировать такие графики по нескольким шаблонам (их примерно десять). Пользователю достаточно ввести данные или указать, где их взять в памяти ПК, а также тип шаблона, и ПК построит нужный график или диаграмму на экране дисплея. Останется на этом графике сделать соответствующие надписи и обозначения.

Пакеты программ табличной обработки

Всем известен способ вычислений на бланке. Напомним его. Пусть А, Б, В, ... — столбцы бланка, а 1, 2, 3, ... — строки. Тогда на их пересечениях находятся клетки, которые можно обозначить по столбцам и строкам: А1, А2, Б3 и т. д. (табл. 1). Если теперь некоторым клеткам задать конкретные значения, то остальные можно вычислять по ним, задавая их зависимость от других в виде формул, например: А2=А1+Б1, Б2=А1—В1, В2=А1×В1. Эти операции и делает ППП табличной обработки. Пользователю следует задать исходные значения клеток или указать, откуда ПК должен взять их, и записать выражения связи между клетками. Остальное выполнит программа.

Например, А и Б — детали узла, 1, 2, 3— свойства, например 1— масса, 2— стоимость, 3— максимальный габаритный размер. Тогда В — узел из этих деталей, свойства которого образуются очевидными формулами: В1=А1+В1, В2=А2+Б2+С, где С — стоимость сборки узла, В3=max(В1, В2), что обозначает максимальное из двух чисел В1 и В2, например max (5, 7)=7.

Таблица 1

	А	Б	В	...	
	1	А1	Б1	В1	...
2	А2	Б2	В2	...	
3	А3	Б3	В3	...	
4	А4	Б4	В4	...	
⋮	⋮	⋮	⋮	...	

Пакеты программ баз данных

Этот вид ППП особенно ценен, если необходимо работать с большим числом всякого рода сведений, т. е. с большими массивами информации, записанных в памяти ПК. Если эта информация строго упорядочена, то ее называют данными. Например, текст статьи не является данными, хотя и содержит много информации. А вот любая заполненная таблица представляет собой данные: табличная структура упорядочивает информацию и превращает

ее в данные. Структурированность данных позволяет довольно просто манипулировать ими. Так появилось понятие *базы данных*. Это совокупность данных, хранимых в памяти компьютера.

Простейшим примером базы данных является кадровая анкета, где каждая позиция данных закрепляется в записи за определенным свойством (его называют обычно атрибутом): фамилия, имя, отчество, год рождения и т. д. При этом отчество для любой записи стоит всегда на третьей позиции, а первым атрибутом во всех записях является фамилия — такова структура этих данных. Располагая такой базой данных, любой администратор может очень быстро получить любую справку о своем работнике. Например, кто из родившихся до 1960 г. в прошлом году получил премию свыше 30% годового оклада. Очевидно, что для этого необходимые сведения (о премировании и т. д.) должны содержаться в базе данных. Ответом на этот вопрос будет перечень работников, обладающих указанными свойствами.

Другим, более «личностным» примером является база данных «записная книжка», каждая запись которой аналогична записям в бумажной книжке, имеющей атрибуты: фамилия, имя, отчество, телефон, адрес и т. д.

Очень распространена база данных, хранящая библиографию работ по интересующему пользователя вопросу. «Копаться» в такой базе данных одно удовольствие: она позволяет отыскивать работы по любому ключевому слову или их набору, что никак нельзя сделать при «старом» бумажном ведении картотеки (на библиографических карточках).

Пакет баз данных позволяет пользователю самому сконструировать свою базу данных, соответствующую тем данным, которыми он располагает. Кроме того, в пакете имеется программа, обеспечивающая пользователю возможность манипулировать созданной базой данных, работать с ней, изменять, пополнять и корректировать данные. Эту программу называют СУБД — Система Управления Базой Данных. Каждый из таких пакетов записан на гибком диске. Здесь же записан небольшой курс обучения правилам работы с этим пакетом и приведены наглядные примеры. Такие гибкие диски с ППП обычно тиражируются, что обеспечивает их дешевизну на рынке.

КОМПЬЮТЕР — СРЕДСТВО ФОРМАЛИЗАЦИИ ЗНАНИЙ

Если попытаться заглянуть вперед, пофантазировать, то ПК представляется инструментом формализации знаний пользователя-профессионала. Действительно, каждый из нас является специалистом в какой-то области знаний, умений, навыков. Как правило, эти знания сугубо индивидуальны и с очень большим трудом поддаются формализации. Очень соблазнительно использовать персональность персонального компьютера для попытки формализации таких знаний и умений, создавая соответствующие программы. Скорее всего такой формализации будут

поддаваться рутинные знания, формализация которых с помощью ПК позволит все внимание обратить на творческую компоненту своей деятельности. Ввиду индивидуального характера таких знаний эту формализацию может осуществить только их владелец. Для этого ПК должен предоставить ему широкие возможности.

Но ... пока совершенно непонятно, как же можно это сделать. А поэтому можно лишь мечтать об автоформализации с помощью ПК и фантазировать. Если действительно это удастся сделать, т. е. удастся создать программы формализации знаний специалиста, то ПК усилит интеллектуальные возможности человека. Это будет совершенно новая форма использования компьютера. ЭВМ всегда считалась средством, расширяющим возможности человека в уже формализованных областях, таких, как вычисления, сортировка, обработка знаковой информации, и др. В случае автоформализации ПК усиливает возможности человека в неформализованной сфере его знаний.

Формализм всегда был границей, отделяющей то, что может делать компьютер, от того, что ему недоступно. Автоформализация по своему замыслу должна стать процессом преодоления этой границы и передачи ПК того, что еще вчера ему было недоступно. И осуществить это, возможно, позволит диалог пользователя-профессионала (в своей области, разумеется) с ПК, оснащенным пока существующими лишь в мечтах программами автоформализации. Появятся такие программы или нет, сейчас сказать трудно, но и мечтать не возбраняется.

Однако передача компьютеру неформальных знаний специалиста происходит уже сейчас. И этим широко пользуются при создании так называемых баз знаний (о них мы поговорим в последней главе). Но делается это не в режиме диалога компьютера и пользователя, а при диалоге специалиста с инженером по знаниям — когнитологом. Это долгий путь, отличный от пока гипотетической автоформализации.

ЧТО ЖЕ ДАЛЬШЕ?

По молчаливо принятому, хотя и неписанному правилу ПК должен удовлетворять принципу трех М (миллионов): миллион операций в секунду по производительности процессора, миллион байт оперативной памяти (ОЗУ) и миллион точек на экране дисплея. Только в этом случае можно реализовать в полной мере те черты, которые сделали ПК первой массовой ЭВМ, революционизирующей методы обработки информации. Но это вовсе не значит, что ПК, не уложившийся в это правило, будет плох. Просто его возможности будут ограничены. Указанное правило следует считать скорее идеалом, к которому нужно стремиться при разработке новых ПК. Дело в том, что каждому, кто сел за пульт ПК, вскоре «приходит аппетит» к производительности компьютера (его

обычно тяготит ожидание результата решения громоздких задач), к емкости ОЗУ и четкости изображения. При выполнении правила трех М этот момент наступит нескоро, а возможно, и вообще не наступит.

Академик А. П. Ершов назвал феномен ПК «скандалом в благородном семействе», который вызвал потрясение основ наших представлений о возможностях и применении вычислительной техники. Чем закончится этот «скандал», сейчас сказать трудно. Ясно, что нам предстоит еще долго ощущать неожиданное воздействие ПК на нашу профессиональную и повседневную деятельность.

Конечно, у этого процесса есть и негативные стороны. Но пока кажется, что их немного и они незначительны. Например, у пользователя, систематически работающего с ПК, наверняка появится зависимость от ПК и его состояния. Все неисправности ПК этот пользователь будет воспринимать очень болезненно, ведь они лишат его возможности нормально работать. Возврат к старым методам для такого пользователя затруднен, а иногда и вовсе невозможен: потребует от него резкого снижения производительности и качества работы (например, экскаваторщик едва ли возьмет в руки лопату, если откажет экскаватор). Это обстоятельство заставляет производителей ПК повышать его надежность. Именно поэтому следует стремиться, чтобы наработка на отказ для ПК была не меньше 5 лет. Это трудно, но возможно.

— А почему бы не больше? — спросил Мегрэ Поля.— Ведь 10 лет безотказной работы лучше, чем 5!

— Но ведь за надежность надо платить! И немало. Надежные схемы — это, прежде всего, дублирование схемы. А чтобы удвоить надежность, надо, грубо говоря, удвоить аппаратуру, т. е. сделать компьютер в два раза дороже. Вот и приходится решать: иногда ремонтировать дешевый компьютер или заплатить дорого за тот, который почти не требует ремонта.

— Это зависит от того, какие задачи я буду на нем решать,— задумчиво сказал Мегрэ.— Если при поломке компьютера я потеряю ценную информацию, то он ломаться не должен!

— А тогда придется раскошелиться!— весело закончил Поль.

Поль распаковывал коробки, насвистывая модный шлягер.

— Никак не могу привыкнуть к современному способу выражения эмоций,— заметил Мегрэ.

— Извините, шеф,— смутился Поль.— Это я собираю компьютер, который нам прислали из префектуры по вашему заказу, и увлекся.

— То есть как, собираете?— изумился Мегрэ.—А где представители фирмы? Это же не холодильник, который можно включить самому.

— Не волнуйтесь, шеф,— улыбнулся Поль.— Это не просто компьютер. Это персональный компьютер. А следовательно, он ориентирован на нас с вами.

Поль быстро соединил шнурами блоки компьютера.

— Вот и вся сборка,— и Поль вставил вилку в розетку.

— Действительно, несложно,— заметил Мегрэ.— Этот монитор на самом деле похож на небольшой телевизор, а клавиатура— очень похожа на пишущую машинку, только тоньше. Это, по-видимому, принтер. Если от пишущей машинки отделить клавиатуру, то и получим принтер, не так ли Поль?

— Почти. В отличие от пишущей машинки здесь можно «печатать» не только на бумаге, но и на экране монитора. И это основной режим работы с компьютером. Вывод же на печать — завершающий этап работы с компьютером.

— А что это такое?— Мегрэ указал на плоский ящик с двумя щелями.

— Это системный блок — центральная часть компьютера. В нем расположен процессор, оперативная память (ОЗУ), винчестерский диск и дисководы двух гибких дисков (дискет). Через эти щели в компьютер вводятся дискетки.

— А почему дисковода два? Ведь, кажется, можно было бы обойтись и одним.

— Почти всегда приходится работать с двумя дискетками. Одна «своя», а другая «чужая». На свою вы записываете всю ту информацию, которая вам понадобится в следующий раз. Это вроде личной записной книжки. А чужая — лишь источник информации, например досье и, прежде всего, пакеты прикладных программ, которыми приходится пользоваться при работе с компьютером. Именно эти пакеты и обеспечивают необходимый сервис и делают компьютер дружественным. Поэтому всякий компьютер всегда сопровождается пачкой дискет с пакетами прикладных программ, среди которых изрядную долю составляют игры.

—Ну, оставим эти игры для детей и внуков.

— Подождите, не торопитесь, комиссар. Игры и нам пригодятся. Ведь это модели, на которых можно отрабатывать навыки, нужные не только детям, но и взрослым. Так, для эффективной работы с компьютером нужно овладевать клавиатурой. Для этого нужно уметь печатать на пишущей машинке — клавиатура компьютера и машинка очень похожи. Есть несколько игр для обучения работе на клавиатуре компьютера. Делается это так. На экране монитора создается игровая ситуация, которую вы можете изменить в свою пользу, набрав определенную комбинацию знаков на клавиатуре. Сначала, пока вы не освоились, для набора компьютер выделяет много времени. Но по мере обучения это время сокращается и стимулирует вас убыстрять реакцию. Так, играя, почти незаметно для себя вы стараетесь и обучаетесь. Есть программы игр в шашки, шахматы, го и др. Это для разрядки.

— Что ж,— заметил Мегрэ,— с удовольствием разок сыграю с компьютером в шахматы. Только едва ли это будет интересно: он играет либо лучше, либо хуже меня и, значит, будет всегда или выигрывать, или проигрывать, что неинтересно.

— На этот случай в шахматной программе (и в других играх тоже) введен параметр силы программы. Это число ходов вперед, которые во время игры просматривает компьютер. Этот параметр вы назначаете сами. Если хотите встретиться с противником послабее (посильнее), то его значение должно быть назначено малым (большим). А для того, чтобы все время играть на равных, повышайте этот параметр при выигрыше и понижайте при проигрыше.

— А зачем нужен этот блок связи?— полюбопытствовал Мегрэ.—Что, этот компьютер может выполнять роль секретарши?

— Нет. Этот блок соединяет ваш компьютер с другим таким же через телефонную сеть. Очень полезная связь, ею можно воспользоваться, если необходимо обменяться компьютерной информацией. Например, для работы с каким-то нужным досье вы должны иметь его в виде дискетки. Чтобы получить ее, вовсе не нужно

посылать курьера в архив префектуры. Достаточно набрать номер телефона и попросить служащего поставить нужную дискетку в свой компьютер. После этого вы и ваш абонент кладете свои телефонные трубки на блоки связи и включаете их: вы на прием, он на передачу. В результате содержимое его дискетки будет передано по телефону и записано на вашу дискетку. Вот и все!

— А смогу я воспользоваться этим устройством как почтой и надежно передавать сообщения? Ведь телефон не всегда надежен.

— Разумеется. Предусмотрен режим электронной почты. В этом случае наши сообщения попадают сразу в память компьютера вашего абонента, а оттуда на экран дисплея (монитора). При желании он может «сбросить» ваше сообщение на принтер и получить его бумажную копию. А что касается надежности, то при помехах сообщение будет повторяться до тех пор, пока не будет принято правильно. Для этого есть специальный режим работы компьютера по программе, записанной на специальной дискете. Так что почти все возможности компьютера определяются его программами, которые можно легко вводить в него с соответствующих дискет.

— Что же, принимаем этот компьютер на вооружение нашей группы. И для начала введите в его память все сообщения, полученные сегодня.

3. КАК БЫ ТЕБЕ ЭТО ОБЪЯСНИТЬ!

(Проблема общения)

Общение человека с компьютером, пожалуй, одна из самых сложных проблем. Уж очень не похожи друг на друга партнеры. С одной стороны человек, а с другой — электронный автомат, пусть даже программируемый, каким является любой компьютер. И это общение происходит на языке, т. е. почти так, как между людьми.

А зачем для общения с компьютером нужен язык? Ведь компьютер не более чем автомат! Ну и будем управлять этим автоматом, как всеми автоматами — с помощью кнопок, рукояток и переключателей. Общение слушателя с радиоприемником или магнитофоном никто не называет диалогом. Почему же для общения с компьютером мы обращаемся к языку?

Прежде всего потому, что компьютер не просто автомат, а автомат для переработки информации, которая, кроме того, часто задается в языковой форме. Для его функционирования надо составить программу переработки этой информации. Разумеется, программа может быть составлена с помощью кнопок и переключателей, но лишь в чрезвычайно простом случае. Именно с этим случаем мы встречаемся при работе с обычным карманным калькулятором, где кнопками указывается операция (сложить, умножить и т. д.), которую нужно произвести над вводимыми числами. Но если вычисления не столь просты и содержат много операций, то при составлении программы необходимо соблюдать опре-

деленные правила — грамматику языка программирования. Заметим, что правила грамматики следует соблюдать и при обращении с карманным калькулятором, хотя они и крайне просты и сводятся к соблюдению порядка введения чисел в калькулятор и выбранной операции (в простейших эта операция расположена между числами, как учат школьников, а в других — перед числами, к которым она применяется,— так проще калькулятору). Это тоже язык, но очень простой. Здесь простота общения получена только потому, что калькулятор мало что умеет. Все его «умения» обозначены кнопками вызова операций на пульте. Если же потребуется решить более сложную задачу, то не обижайтесь на калькулятор: он на это не рассчитан. Обратиться придется к компьютеру и притом на более сложном языке.

Дело в том, что язык как средство общения человека с компьютером зависит от тех задач, которые предстоит решать компьютеру. Чем сложнее задача, тем выразительнее должен быть язык, на котором формулируется задание и описывается способ решения задачи. Сложность языка общения зависит от сложности решаемой задачи. А так как на компьютер мы возлагаем решение сложных задач, то и язык общения с ним должен быть сложным.

Здесь следует различать два случая. Компьютер умеет решать ряд задач — в его памяти есть программы решения, и проблема общения состоит в том, чтобы «растолковать» ему, какую именно задачу следует решать. Другой случай — компьютер не имеет такой программы и ее надо составить, т. е. указать компьютеру не только что решать, но и как решать. Стоит ли говорить, что во втором случае язык общения должен быть более сложным, выразительным, чем в первом.

Для установления надежной коммуникации на языке между двумя объектами каждый из них должен понимать этот язык, иначе коммуникации не получится. Если одним объектом является компьютер, а другим — человек, обязанность «понимать» ложится тяжелым бременем на компьютер — ведь это только автомат. В диалоге человек — компьютер партнеры неравноправны. Человек всегда поймет компьютер (для этого, правда, ему нужны знания и время), а вот компьютер далеко не всегда понимает человека. Именно человеку приходилось все «разжевывать» первым компьютерам на их языке.

При разработке языков для общения человека и компьютера всегда приходится делать выбор: кому облегчить работу — человеку или ЭВМ? Ведь язык, который «понимает» ЭВМ, слишком далек от человеческого — это язык нулей и единиц. Чтобы «говорить» на нем, программист должен знать устройство компьютера и иметь навык общения с ним. Если же язык общения близок к человеческому, то для «перевода» с него компьютеру потребуется много времени, большая емкость памяти и, что самое

трудное, предварительно создать программу преобразования фраз человеческого языка на машинный (об этой еще не полностью решенной проблеме мы поговорим в гл. 9).

Первые компьютеры работали относительно медленно, и память их была невелика. Поэтому эта дилемма, естественно, была решена в пользу машины — человек просто должен был научиться программировать на машинном языке.

Что же это такое?

МАШИННЫЙ ЯЗЫК

Он и по сей день остается для компьютера основным, самым близким и понятным ему. Это язык команд, которые может выполнять данная машина. Очевидно, что машинный язык определяется устройством и схемой компьютера. Каждый новый компьютер порождает новый машинный язык. Например, команда «Сложить два числа, хранящиеся в ячейках памяти А и В» записывается в виде трех элементов КАВ, где К — код (номер) операции «сложить», А и В — адреса (номера) ячеек памяти (ОЗУ), где хранятся слагаемые (операнды). Аналогично может быть записана любая другая команда над двумя числами (операндами), которые хранятся по адресам А и В. Современные компьютеры имеют большую и разветвленную систему таких команд. Одних только сложений более десятка: сложение содержимого ячеек А и В, когда А и В — номера регистров, когда А — номер регистра, а В — номер ячейки ОЗУ и т. д. Например, машины серии ЕС ЭВМ содержат около 200 различных команд.

Но одних команд для работы ЭВМ мало. Так как ЭВМ сама «не знает» никаких чисел, то все требуемые программой числа (их называют константы) должны быть введены в память вместе с программой (это называется описанием констант). Например, при необходимости умножить что-либо на 5 в программе должен быть указан адрес ячейки памяти, где хранится число 5, которое следует занести туда. При обращении к этой ячейке константа становится операндом, т. е. числом, с которым работает оператор.

Программа для ЭВМ, написанная на машинном языке, представляет собой длинную колонку команд, каждая из которых состоит из трех элементов-чисел: номера (кода) оператора и адресов первого и второго операндов, над которыми производится операция. А куда девается результат выполнения команды? Это зависит от оператора К. Для большинства операторов результат отправляется по первому адресу (А), для других — в специальный регистр (таких регистров обычно несколько, они являются очень удобным хранилищем промежуточных результатов).

Например, вот так выглядит программа решения задачи сложения двух чисел, расположенных в ячейках ОЗУ с номерами 125 и 168, когда результат

следует занести в ячейку под номером 193:

88	6	0125	Загрузить в регистр 6 содержимое ячейки 125. Здесь 88— код оператора «загрузить», 6 — номер регистра, куда загружается число из ячейки 125, 0125 — номер ячейки
90	6	0168	Сложить содержимое регистра 6 с содержимым ячейки 168. Здесь 90 — код оператора «сложить». Результат сложения по коду 90 записывается в регистр первого слагаемого, т. е. в регистр 6 (есть другие коды сложения, при которых результат направляется в другое место, например по адресу второго слагаемого)
80	6	0193	Записать содержимое регистра 6 в ячейку 193. Здесь 80— код оператора «записать в ОЗУ».

Так устроена *двухадресная система команд*. Ее двухадресность определяется числом адресов в команде. Количество адресов ячеек памяти, которые можно упоминать в одной команде, называют *адресностью* компьютера. Двухадресная система команд наиболее распространенная, но далеко не единственная. Существуют еще одноадресная, трехадресная и даже четырехадресная системы команд.

Одноадресная команда имеет структуру КА, где К — код операции и А — адрес одного из операндов, другой при этом должен находиться в специальном регистре — накопителе. Результат же заносится в тот же регистр-накопитель. Очевидно, что нужный операнд в этот накопитель следует заносить всегда предварительно. На это есть специальная команда.

Трехадресная команда имеет структуру КАВС, где К — по-прежнему код операции, А и В — адреса операндов, а С — адрес ячейки, куда следует поместить результат.

В *четырехадресной команде* KABCD добавляется еще четвертый адрес D, где находится следующая команда, которую нужно выполнить в соответствии с программой. Заметим, что одно-, двух- и трехадресные машины имеют естественный порядок выполнения команд — последовательный, так, как они расположены в программе. Четырехадресная машина позволяет располагать команды произвольно, что иногда бывает очень важно.

«ЗА» И «ПРОТИВ» МАШИННОГО ЯЗЫКА

Чтобы программировать на машинном языке, следует знать не только всю систему команд той ЭВМ, для которой пишется программа, но ее и устройство. Это нужно прежде всего для того, чтобы представлять, как будет она реагировать на ту или иную команду. Без этого программы не напишешь! Следует также следить, чтобы не «затереть» нужную информацию в памяти

при отсылке результата в ОЗУ. Ведь при этом уничтожается прежнее содержимое ячейки, куда записывается новое значение.

Поэтому так трудно программировать на машинном языке. Но зато такая машинная программа имеет одно неоценимое достоинство — она точно описывает весь вычислительный процесс переработки информации данной ЭВМ при решении поставленной задачи. Располагая машинной программой, всегда можно узнать состояние всех блоков ЭВМ в любой момент времени. А это очень важно при разработке и усовершенствовании самой ЭВМ, а также при составлении оптимальных программ, которые отличаются тем, что занимают минимальный объем памяти или позволяют получить результат за кратчайшее для данной ЭВМ время.

И еще. Для понимания машиной такой программы ничего не нужно, ведь написана она на языке самой ЭВМ. Здесь трудно человеку, составляющему программу. Он должен быть высококвалифицированным программистом (недаром лет 40 назад за составление программы, решающей новую важную задачу, можно было получить ученую степень). И, вообще, в языковых взаимоотношениях человека и козпьютера имеет место довольно неприятный дуализм: чем легче одному, тем труднее другому. На стадии программирования на машинных языках все трудности ложатся на плечи человека. Но зато он получает в свое распоряжение все возможности данной ЭВМ: только с помощью машинного языка можно реализовать все особенности конкретной ЭВМ, а следовательно, написать самую лучшую программу, которую только можно написать для этой машины, другими словами, самую короткую, самую быструю, самую простую и т. п. Все это порознь, разумеется, все «самое» в одном совместить нельзя нигде и никогда, в том числе и в программировании.

БЛИЖЕ К ЧЕЛОВЕКУ!

Трудности программирования на машинном языке настолько велики, что появилась естественная идея писать программу на языке, более простом, удобном для человека, а потом преобразовывать ее с помощью той же ЭВМ в программу на машинном языке. Ведь ЭВМ — машина для преобразования информации, и грех было бы не воспользоваться такой возможностью. Задача, таким образом, сводится к переводу программы, написанной на языке, удобном для человека, в программу на машинном языке.

Но здесь возникла серьезная проблема автоматизации перевода с одного языка на другой. Скажем сразу, что в общем случае (для естественных языков, таких, например, как русский и английский) эта проблема не решена и, возможно, в принципе не разрешима. Мы, люди, понимаем друг друга не только потому, что знаем язык, но и потому, что живем в одном и том же мире.

ЭВМ же «живет» в другом машинном мире, чуждом «страстей человеческих». И основным препятствием является неоднозначность понятий, используемых человеком. Конкретная ситуация определяет смысл понятия. Именно поэтому ЭВМ не может выступать в качестве хорошего переводчика с одного языка на другой (по крайней мере, в ближайшее время). Но формальные языки ЭВМ переводить может, они являются представителями машинного мира и поэтому совершенно однозначны и непротиворечивы. Об этом позаботились их создатели.

Процесс преобразования программы, написанной на одном языке, в программу, написанную на другом языке, называют *трансляцией*, а программу, выполняющую эту функцию,— *транслятором*. Легко заметить, что транслятор выполняет функцию автоматического переводчика с одного формального языка на другой, тоже формальный. Стоит ли говорить, что программа такого транслятора очень сложна и создают ее программисты высокого класса. Очевидно, что выходным языком транслятора всегда является уже известный нам машинный язык. А входным?

СИМВОЛИЧЕСКИЕ ЯЗЫКИ

Исторически первыми входными языками для трансляции в машинные стали символические языки. Суть их заключается в том, что коды операций и адреса операндов записываются в символической форме, удобной и понятной человеку. Например, уже известная нам двухадресная команда КАВ может быть записана на символическом языке в виде СЛЖ А X1, где СЛЖ — удобный символ операции «сложить», а А и X1 — символы, т. е. обозначения, имена (а не номера) адресов первого и второго слагаемого. Содержимое ячеек памяти с этими адресами и будет складываться, а результат будет занесен в ячейку с адресом А.

В символических языках появляются переменные. Это названия (имена) тех ячеек памяти, где расположены интересующие пользователя величины. Именем (или символом) может быть любая последовательность букв и цифр, начинающаяся с буквы. Например, A, B, X1 являются именами адресов, где хранятся переменные a, b и x_1 соответственно, а переменным φ_2, $\varkappa_4$ удобно дать имена FI2 и КАРА4. Эти имена и будут обозначать ячейки памяти, где хранятся конкретные значения переменных φ_2 и $\varkappa_4$ в данный момент. В другой момент их значения могут измениться, но имена ячеек остаются неизменными. Таким образом, каждая переменная имеет свою ячейку памяти, где и хранится ее значение. Об этом заботится транслятор, на который возлагается функция перевода имени переменной в номер ячейки, содержащей значение этой переменной.

Легко заметить, что имя переменной эквивалентно номеру ячейки памяти (ОЗУ), где хранится значение этой переменной,

но имя для человека значительно удобней: он всю свою жизнь оперирует названиями, именами вещей, его окружающих. Составляя программу на символическом языке, программист оперирует только именами переменных, а в машинной программе транслятор вместо каждого имени поставит адрес ячейки памяти, где хранится значение соответствующей переменной. Для этого в начале программы всегда указываются все переменные и требуемый для них объем ячеек памяти (и все константы, как в машинном языке).

Приведем пример программы на символическом языке для решения задачи суммирования двух чисел $x_3 = x_1 + x_2$:

ЗГР	РЕГ6	X1	Загрузить в регистр 6 переменную x_1. Здесь ЗГР — символ (имя) оператора «загрузить», РЕГ6 — символ регистра 6, X1 — символ переменной x_1, хранящейся в ячейке 125 ОЗУ (см. предыдущий пример).
СЛЖ	РЕГ6	X2	Сложить содержимое регистра 6 с переменной x_2. Здесь СЛЖ — символ оператора «сложить», X2 — символ второй переменной x_2, хранящейся в ячейке 168 ОЗУ
ЗПМ	РЕГ6	X3	Запомнить содержимое регистра 6 в ячейке, соответствующей переменной x_3. Здесь ЗПМ — символ оператора «запомнить», X3 — символ третьей переменной x_3, хранящейся в ячейке 193 ОЗУ

Как видно, символический язык оперирует в основном символами. Такой язык, по существу, является машинным языком, но с удобной мнемонической символикой, которая легко запоминается, и поэтому значительно удобнее для программиста, чем цифры машинного языка. (Заметим, что цифры тоже являются символами, но они неудобны для человека. Именно это и побудило создавать символические языки с символами-именами.)

Транслятор с программы символического языка на машинный достаточно прост. Ему лишь нужно заменить символические обозначения цифрами номеров кодов и ячеек памяти. В приведенном примере транслятор сделает такие замены: СЛЖ на 90, ЗПМ на 80, X1 на 0125 и тш д. Именно поэтому его часто называют *автоматическим кодировщиком,* сам же символический язык — *автокодом.* Но главное достоинство автокода — автоматизация распределения памяти компьютера. Программисту теперь не надо следить, как распределяется память, и переживать, что будет «затерта» нужная информация.

Транслятор с символического языка часто называют *ассемблером* (англ. assemble — собирать, монтировать), а сам язык — *языком ассемблера.* Так выстраивается цепочка (рис. 2), где программисту стало уже значительно легче, а для преобразования его программы в машинную введен транслятор — ассемблер.

Однако пользователя компьютера обычно интересует лишь



Рис. 2

решение своей конкретной задачи, причем без больших затрат времени и средств. Для него компьютер не более чем инструмент для достижения своих целей, как топор и пила для плотника. (В этом сравнении нет большого преувеличения. Просто к топору и пиле мы привыкли, а к компьютеру еще нет. Лет через сто, желая подчеркнуть простоту какого-нибудь вопроса, возможно, скажут: ...как компьютер.)

Желая применить какой-то новый инструмент, мы прежде всего знакомимся с инструкцией для его применения. Такой инструкцией для пользователей ЭВМ является описание языка программирования. Очевидно, что сколько типов ЭВМ, столько (по крайней мере) и символических языков, так как каждый такой язык ориентирован на систему команд конкретного типа ЭВМ. Именно поэтому их называют *машинно-ориентированными языками*. Очевидно, что это не всегда удобно, а точнее, всегда неудобно. Пользователю часто совершенно безразлично, на какой ЭВМ будет решаться его задача. Так возникла идея языков программирования, не зависящих от ЭВМ, на которой будет решаться задача.

ЯЗЫКИ, НЕ ЗАВИСЯЩИЕ ОТ КОМПЬЮТЕРА

Это языки *высокого уровня*. Отличительной их чертой является независимость от компьютера, на котором будет решаться задача. Поэтому их часто и называют *машинно-независимыми языками*. Пользователь, составляющий программу на языке высокого уровня, совершенно не обязан знать тот компьютер, на котором будет решаться его задача. Ему достаточно знать только символику и грамматику такого языка. Символы операторов этого языка обычно максимально приближены к естественному

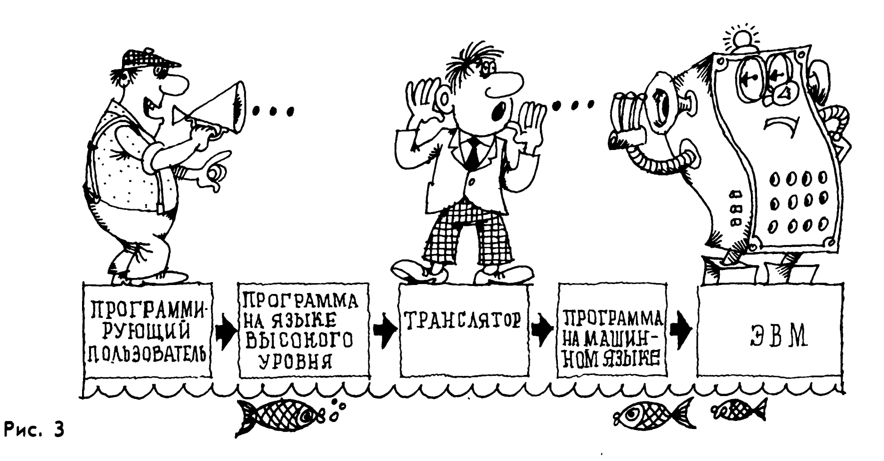


Рис. 3

языку, а грамматика — к правилам манипулирования в той проблемной области, для которой создан этот язык.

Транслятор преобразует эту программу в машинную, которая и выполняется компьютером. Очевидно, что транслятор построен с учетом специфики конкретной машины. Нет транслятора вообще, а есть трансляторы с определенного языка на машинный язык определенной ЭВМ. Схема взаимодействия пользователя с ЭВМ на языке высокого уровня показана на рис. 3.

Простота и доступность этих языков позволяет пользоваться ими широкому кругу специалистов (так, основным правилам простейшего из этих языков Бейсика можно научиться за час-полтора). В языках высокого уровня не надо описывать переменные и константы — это делает сам транслятор. Операторы таких языков ориентированы на специфику решаемых задач. В языках для вычислений имеются, например, такие операторы.

Присваивание, с помощью которого определяется новое значение переменной. Этот оператор имеет вид «:=» и ставится между переменной и ее новым значением. Например, X:=3 (здесь переменной x присваивается значение 3), Y:=Z+U (значение y должно стать равным сумме значений переменных z и u), G:=X*Y (переменной g присваивается значение произведения xy, здесь * — знак умножения). Заметим, что в некоторых языках высокого уровня этот оператор записывается в виде обычного знака равенства «=».

Например, программа задачи сложения двух чисел $x_3=x_1+x_2$, рассмотренная выше в машинных и символических кодах, на языке высокого уровня выглядит очень просто:

X3=X1+X2

и пользователю не нужно беспокоиться о загрузке переменных в регистры — это сделает сам транслятор. Если значение пере-

менной b нужно увеличить на 5, то это записывается в несколько странном виде:

$$B=B+5$$

Это не уравнение: здесь знак равенства является оператором присваивания. В левой части этого выражения стоит новое значение переменной b, а в правой — старое, предыдущее значение.

Как видно, оператору присваивания может предшествовать процесс вычисления по формулам, программирование которых не представляет труда. Например, одно из решений квадратного уравнения $ax^2+bx+c=0$ реализуется (если $b^2-4ac>0$) одним оператором

$$X=(-B+(B**2-4*A*C)*0.5)/(2*A),$$

где ** — знак возведения в степень, а / — знак деления.

Условный переход определяет ветвление программы в зависимости от сложившейся ситуации. Например, оператор ЕСЛИ X> 5 ТО 36 означает, что при $x\leqslant 5$ следует выполнять очередной оператор программы, а при $x>5$ надо обращаться к оператору под номером 36 или с меткой 36. (Меткой называется значок, стоящий перед оператором, к которому следует переходить при появлении этой метки в программе, меткой могут быть любые числа, буквы, значки и т. д.) Наличие оператора условного перехода позволяет делать программы очень гибкими и решать чрезвычайно сложные задачи.

Цикл позволяет повторить однотипные вычисления при изменяющихся значениях переменных. Например, оператор ДЕЛАТЬ X=X*I ДЛЯ I=1 ДО N вычислит значение факториала $N!$ (где N — конкретное число), если предварительно присвоить X=1. Действительно, на первом шаге $x=x\cdot 1=1$, так как вначале $x=1$ и $I=1$. На втором шаге $x=1\cdot 2=2$, так как $I=2$, третьем $x=2\cdot 3$ и т. д. до $I=N$. В результате получим $x=1\cdot 2\cdot 3\ ...\ N=N!$

Далее (в гл. 5) мы рассмотрим основные черты наиболее распространенных языков высокого уровня, таких, как Бейсик, Фортран, Алгол, Паскаль, и др. А всего существует более 3 тыс. (!) языков высокого уровня. Не следует смущаться этим обстоятельством, алгоритмические языки похожи друг на друга. Так, например, Бейсик очень похож на Фортран, а Паскаль на Алгол. Так что хорошее знание одного языка очень облегчает понимание других (как и в естественных языках).

КОМПИЛЯЦИЯ И ИНТЕРПРЕТАЦИЯ — РЕЖИМЫ ТРАНСЛЯЦИИ

При использовании программы, написанной на любом немашинном языке (символическом или языке высокого уровня), процессу решения задачи предшествует процесс ее трансляции в машинную программу. Но сам процесс трансляции может протекать по-разному.

Можно сначала перевести всю программу на машинный язык и только после этого ее выполнять. Этот режим называют *компиляцией*, подчеркивая функциональную идентичность обеих программ — исходной на языке высокого уровня и полученной машинной.

Но часто удобней поступить иначе: транслировать лишь небольшую часть программы и сразу выполнить ее, далее снова транслировать следующий небольшой кусок и снова его выполнить и т. д. При этом трансляция и выполнение программы чередуются: транслируются один или несколько операторов программы, которые немедленно выполняются компьютером. Этот способ называют *интерпретацией*, а программу, реализующую его, *интерпретатором*. Разница между компилятором и интерпретатором аналогична разнице между обычным переводчиком текста и синхронным переводчиком (устной речи). Переводчик текста преобразует текст, создавая новый на другом языке. Синхронный переводчик переводит каждую фразу после ее произнесения.

Режим интерпретации очень удобен при работе с малыми компьютерами, память которых невелика, и поэтому нельзя выделить большой объем памяти для размещения всей машинной программы. Удобна интерпретация и при диалоговом составлении программ, когда нужно быстро убедиться в правильности работы очередного куска программы. Однако если необходимо повторно выполнить программу, но, например, с другими данными, ее снова нужно интерпретировать в машинные коды, на что, естественно, тратится дополнительное время. При компиляции этого не происходит — повторно запускается уже готовая машинная программа (программа на машинном языке), которая хранится в памяти компьютера.

Как видно, процесс составления программы для компьютера требует определенных знаний и навыков (иногда достаточно глубоких). Значит ли это, что диалог с компьютером доступен лишь тем, кто владеет языками программирования? Разумеется, нет!

ПРОГРАММИРОВАНИЕ БЕЗ... ПРОГРАММИРОВАНИЯ

Всех собеседников компьютера можно подразделить на три большие группы:

системные программисты (их меньшинство) — создают программы, облегчающие труд других программистов, например трансляторы, операционные системы (о них в следующей главе);

программирующие пользователи — пишут программы для себя (точнее, для решения своих задач);

непрограммирующие пользователи (их большинство) — хотят решать свои задачи на ЭВМ, не желая знать языки программирования; их часто называют конечными пользователями.

Но возможно ли такое? Оказывается возможно, если в памяти ЭВМ уже достаточно много программ. Нужно лишь найти ту или те, с помощью которых может быть решена ваша задача и воспользоваться ими. Для этого применяется *режим меню* (мы о нем уже упоминали, описывая работу «мыши» в персональном компьютере). В этом случае ЭВМ (точнее, ее программа) выступает вашим активным партнером: задает вопросы и предлагает варианты ответов (это и есть меню), а вы отвечаете номером ответа, который выбираете, или значением запрашиваемого параметра. Вот и все! Самое трудное в таком диалоге дать понять компьютеру, что же вам нужно: решить уравнение, обучиться иностранному языку, получить справку, рассчитать фильтр, подобрать транзистор или какой-либо другой элемент для своей схемы или что-то другое. Как только в процессе диалога задача будет «понята», компьютер из своей памяти достанет программу решения вашей задачи, решит ее и выдаст ответ.

Приведем пример диалога в режиме меню. Пусть вам понадобилось решить квадратное уравнение $ax^2+bx+c=0$, а формулу решения вы забыли (или просто не знали). Компьютер поможет вам в этом.

Пользователь (П) набирает свой текст на клавиатуре и контролирует его на экране дисплея, а компьютер (К) выводит свои реплики на тот же экран.

Этот диалог будет иметь примерно такой вид.

П: ХОЧУ РЕШИТЬ УРАВНЕНИЕ

(Компьютер не понял — в режиме меню он понимает лишь ответы на заданные им же вопросы, поэтому сообщает на экране дисплея о своих возможностях, то, что он умеет делать.)

К: ИМЕЮТСЯ ПРОГРАММЫ ПО СЛЕДУЮЩИМ ТЕМАМ:

1. РЕДАКТИРОВАНИЕ ТЕКСТА
2. РЕШЕНИЕ УРАВНЕНИЙ
3. СПРАВОЧНЫЕ ТАБЛИЦЫ
4. ОБУЧЕНИЕ ИНОСТРАННОМУ ЯЗЫКУ

и т. д.

НАБЕРИТЕ НОМЕР НУЖНОЙ ТЕМЫ

П: 2 (это и есть выбор пользователя из предложенного меню)

К: ИМЕЮТСЯ ПРОГРАММЫ РЕШЕНИЯ СЛЕДУЮЩИХ УРАВНЕНИЙ:

1. АЛГЕБРАИЧЕСКИЕ
2. ДИФФЕРЕНЦИАЛЬНЫЕ

НАБЕРИТЕ НУЖНЫЙ НОМЕР

Если пользователь затрудняется в выборе, например, потому, что не знает разницы между алгебраическими и дифференциальными уравнениями, то ему достаточно задать вопрос, т. е. набрать знак «?», и компьютер объяснит ему, что есть что и приведет примеры. Это вовсе не означает, что компьютер знает больше, чем пользователь. Просто в его памяти на этот случай записан соответствующий текст, составленный тем, кто делал эту программу-меню.

П: 1 (он выбрал правильно, ему действительно надо решить алгебраическое уравнение).

К: УРАВНЕНИЕ
1. ЛИНЕЙНОЕ
2. НЕЛИНЕЙНОЕ
НАБЕРИТЕ НУЖНЫЙ НОМЕР

(Это нужно компьютеру: методы решения линейных и нелинейных уравнений различны, а следовательно, в его памяти находятся разные программы.)

П: 1

(если ответ на вопрос вызывал бы затруднения у пользователя, то в ответ на «?» компьютер пояснил бы, чем отличаются линейные и нелинейные уравнения, т. е. извлек бы из своей памяти тексты, составленные заранее на этот случай, и построил бы вопрос.)

К: КАКОВ ПОРЯДОК УРАВНЕНИЯ?
1. МЕНЬШЕ ИЛИ РАВНО 3
2. БОЛЬШЕ 3

(Дело в том, что алгебраические уравнения до третьего порядка решаются точно и в памяти компьютера есть эти программы. При необходимости решения уравнения выше третьего порядка следует воспользоваться численным методом).

П: 1 (так как уравнение, которое следует решить, квадратное)

К: УКАЖИТЕ ПОРЯДОК УРАВНЕНИЯ

(Если до этого пользователь выбирал один из вариантов ответов, предложенных самим компьютером, то сейчас нужно ответить числом).

П: 2

К: ВАШЕ УРАВНЕНИЕ ИМЕЕТ ВИД
A*X**2+B*X+C=0?

(это машинная запись уравнения $ax^2+bx+c=0$)

ОТВЕТЬТЕ «ДА» ИЛИ «НЕТ».

П: ДА

(Если бы ответом было «нет», то компьютер стал бы выяснять, где вами была допущена ошибка и снова задавал бы предыдущие вопросы.)

К: ВВЕДИТЕ ДАННЫЕ
A=
B=
C=

П: вводит конкретные значения коэффициентов a, b и c своего уравнения.

К: X1=...; X2=...;

(Это ответ, где точками обозначены конкретные значения корней. Он был получен путем решения уравнения по известной школьной формуле, программа вычисления которой была в памяти компьютера и приведена в разделе «Языки, не зависящие от компьютера» в этой главе).

К: ВЫ УДОВЛЕТВОРЕНЫ?
ОТВЕТЬТЕ: «ДА» ИЛИ «НЕТ»

(Результат может не удовлетворить пользователя, например, когда корни окажутся мнимыми, а должны быть действительными. Ему придется либо пересмотреть вывод этого уравнения — для этого компьютер не нужен, либо изменить уравнение. Пусть случилось последнее).

П: НЕТ

К: ЧТО ИЗМЕНЯЕТСЯ?

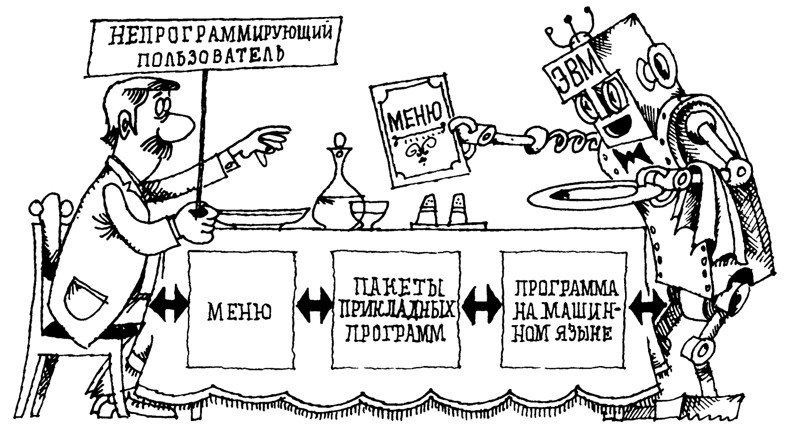


Рис. 4

1. ТЕМА
2. ТИП УРАВНЕНИЯ
3. СТРУКТУРА УРАВНЕНИЯ
4. ПОРЯДОК
5. КОЭФФИЦИЕНТЫ

НАБЕРИТЕ НУЖНЫЙ НОМЕР

Пользователь может вернуться к разным этапам диалога, даже сменить тему — не решать уравнения, а, например, редактировать с помощью компьютера текст своей будущей статьи. Если же пользователь продолжает интересоваться решением своего уравнения, то он должен указать номер этапа, начиная с которого в уравнении произойдут изменения. При этом данные, полученные компьютером до этого этапа, будут храниться в памяти для продолжения диалога.

Как видно, диалог в режиме меню прост и для пользователя (ему не надо программировать), и для компьютера. Действительно «интеллект» компьютера здесь невелик. Ему для понимания пользователя достаточно различать цифры — номера ответов и слова «да» и «нет». Остальное запрограммировано заранее и хранится в его памяти.

Схема взаимодействия пользователя с ЭВМ в режиме меню показана на рис. 4. Здесь компьютеру необходимо иметь программу меню с мощной поддержкой памяти, где хранятся все варианты вопросов и ответов, чтобы обеспечить все возможные варианты диалога с пользователем, и пакет программ, среди которых обязательно должна быть та, которая решит задачу пользователя.

Как видно, «язык» меню самый простой и поэтому самый предпочтительный для непрограммирующего пользователя и, несомненно, перспективный для массового применения ЭВМ. Но его можно использовать далеко не всюду, а лишь в довольно узкой области знаний.

КОГДА МЕНЮ «НЕВКУСНОЕ»

Дело в том, что для работы в режиме меню требуется заранее предвидеть все ответы пользователя на все вопросы компьютера. Если на каждый вопрос иметь только два ответа (меньше нельзя), то для всех случаев N-шагового диалога (один шаг: вопрос-ответ) необходимо не менее 2^N вопросов — за каждым ответом пользователя должен следовать вопрос компьютера с не менее чем двумя вариантами ответа. При $N=10$ нужно иметь более тысячи таких вариантов ($2^{10}\approx 1000$), а при $N=50$ свыше 10^{15} вопросов — это миллион миллиардов. Здесь проблема не столько в том, чтобы разместить такой гигантский объем информации (емкость ОЗУ современных компьютеров значительно меньше — 10^9—10^{10} байт), сколько в том, чтобы записать такое число вопросов. Несложный подсчет показывает, что даже произносить эти вопросы придется свыше 10 млн. лет, если на один вопрос тратить всего 1 с. Это и ограничивает возможности метода.

Именно поэтому далеко не для всякого диалога с компьютером можно воспользоваться методом меню. Как же быть? В этом случае приходится обращаться к языку сверхвысокого уровня — естественному языку, на котором можно описать любую задачу, или (на худой конец) языку, близкому к естественному. Схема такого взаимодействия показана на рис. 5.

Здесь компьютер должен не только понимать естественный язык пользователя (эта функция возлагается на программу лингвистического процессора), но и создавать программу решения задачи пользователя (это делает программа «планировщик», который может пользоваться готовыми программами из имеющихся в памяти пакетов прикладных программ). Отличительной чертой этой схемы является наличие баз знаний о языке пользователя (для лингвистического процессора) и о предметной области, в

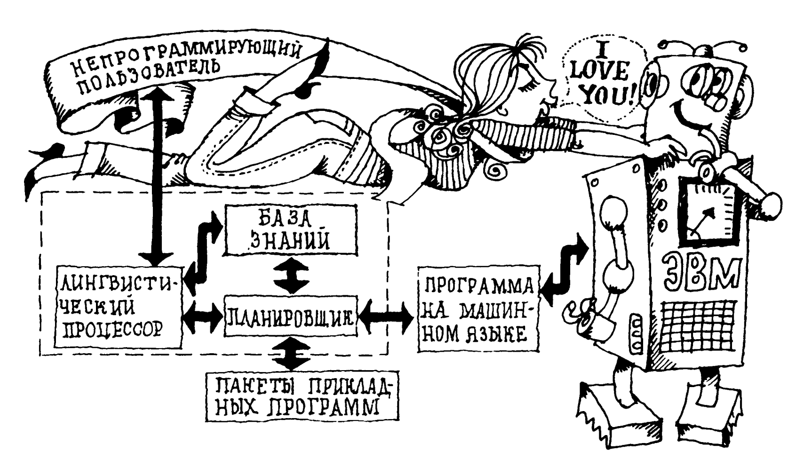


Рис. 5

которой работает пользователь. Эта база знаний нужна еще и на тот случай, если для задачи пользователя не найдется программы ее решения в пакете прикладных программ. Тогда планировщик составит нужную программу, оперируя знаниями о предметной области пользователя. Такие программные системы называют *интеллектуальным интерфейсом* (он выделен на рис. 5 штриховой линией; подробнее о нем см. в гл. 11).

Таким образом, процесс общения человека с компьютером одновременно и очень сложный, и «простой». Его сложность связана с необходимостью создания и функционирования больших и громоздких программ трансляторов, меню, лингвистических процессоров и планировщиков. А простота должна быть внешней, когда удается «упрятать провода вовнутрь», чтобы пользователь видел в компьютере не автомат, а добросовестного и заботливого помощника.

— Как далеко все это от нашей реальной жизни,— задумчиво сказал Мегрэ.— И вы, Поль, всерьез надеетесь, что такими средствами можно что-то существенно изменить?

— Что вы имеете в виду?

— Да хотя бы решение уравнений. Как часто вам, Поль, приходилось решать уравнения?

— Очень давно. Но приведенный здесь диалог лишь иллюстрирует возможности метода меню. Просто автор — инженер и, естественно, привел примеры из своей практики, когда он забыл, как решать квадратное уравнение. Этот пример нагляден и поучителен для всех, кто попадает в подобную ситуацию.

— А я вот комиссар полиции!—раздраженно буркнул Мегрэ.—И для меня такие примеры все равно, что слону кинематограф: мелькает что-то, а съесть нельзя!

— А вы бы хотели, чтобы автор привел пример из вашей следственной практики?— ехидно заметил Поль.

— Да нет,— смутился Мегрэ.—Но все-таки лучше, если пример будет не математическим, если такой пример автор вообще может привести.

— Ваш сарказм здесь ни к чему. Автор действительно инженер, и ему трудно угодить каждому читателю, который хочет немедленно использовать компьютер в своей работе. Тем более, что профессиональное применение компьютера требует помощи профессионалов.

— Это что, я должен сам составлять программы? Так следует вас понимать?— изумился Мегрэ.

— Да. Именно так,— твердо сказал Поль.— До реального программирования вас не допустят, а вот дать необходимую информацию для программистов вам придется. Чтобы компьютер смог помочь вам в раскрытии преступления, он должен знать и уметь не меньше вас. А откуда ему взять все это? Только от вас! Вот и получается, что при составлении программ для компьютера — помощника сыщика —без вас не обойтись. Обо всем этом автор написал целую главу (гл. 14), где описаны экспертные системы.

— Это компьютерные системы, основанные на данных, полученных от эксперта-профессионала,— догадался Мегрэ.— Так зачем же откладывать на самый конец то, что нужно прежде всего?

— Во-первых, экспертные системы — это самые молодые компьютерные системы, они появились совсем недавно. Поэтому и описаны они в самом конце книги — таков закон жанра научной популяризации. А главное то, что эти системы используют все современные достижения компьютерной техники и теории искусственного интеллекта, о которых рассказано в этой книге. Так что, если хотите разобраться в экспертных системах, нужно еще многое узнать.

— Ну что ж,— уныло заметил Мегрэ,— придется читать дальше. И все-таки неужели в работе каждого следователя нет того, что можно было бы уже сейчас передать компьютеру?

— Ого, шеф! Вы становитесь поклонником компьютерной криминалистики. Конечно, есть. Это прежде всего выполнение инструкций, предписывающих любому следователю определенную форму поведения, чтобы не нарушать закон и использовать тот огромный опыт, который накопила криминалистика за свою историю. Например, протокол осмотра места происшествия. Помните как бедняга Жак обливался потом, вспоминая порядок осмотра, предусмотренного инструкцией, и наверняка что-то пропустил. А ведь это типичная программа-меню, которая ведет следователя, не позволяя ему пропустить что-либо, необходимое для следствия. Например, на слово «труп» такая программа немедленно задаст целую серию вопросов, ответы на которые необходимы всякому следствию, и ничего не упустит.

— Так уж ничего!— усмехнулся Мегрэ.— Иной раз при осмотре увидишь такое, что никак нельзя уложить в самую хитрую подробную инструкцию.

— Разумеется. Здесь речь идет о выполнении рутинной операции — выполнении инструкции. Если же у вас есть какие-то специальные соображения по поводу осмотра, то добавляйте их к тому, что предписывает программа инструкции. Этим вы только улучшите описание.

— Что ж,— задумчиво произнес Мегрэ,— в этом есть свой смысл. Рутинные операции программируются методом меню. А нерутинные?

— ... программируются путем составления своей программы на языке высокого уровня,— усмехнувшись, закончил Поль,— т. е. являются теми дополнительными замечаниями, которые сопровождают каждый полицейский протокол.

— И самым трудным пунктом для начинающих сыщиков, так как эти замечания невозможно оттранслировать на человеческий язык,— засмеялся Мегрэ.

4. «ДУША» КОМПЬЮТЕРА
(Операционная система)

Мы неоднократно говорили, что компьютер — это автомат для переработки информации. Хоть и программируемый и очень сложный, но все же автомат. Именно так и относятся к нему до тех пор, пока не поработают с ним некоторое время, особенно с персональным компьютером. Достаточно посидеть немного за дисплеем компьютера, как отношение к нему невольно изменяется. Дело в том, что процесс общения современного компьютера с вами происходит на языке, приближенном к естественному. Его вопросы, замечания, указания, реплики вполне ос-

мысленны и, хочется сказать, разумны! И, наверняка зная, что это автомат, вам кажется, что вы общаетесь с разумным существом. Пусть не всегда понятливым, иногда просто бестолковым, но все же разумным и исполнительным собеседником. Это ощущение одушевленности компьютера создает его операционная система, которую без большой натяжки можно назвать душой компьютера. Если процессор и память выполняют роль мозга, а дисплей — лицо компьютера, то операционную систему естественно считать его душой.

Но появление ее вовсе не связано с желанием «одушевления» компьютера. Никто и никогда не ставил своей целью вселить «душу» в его электронные внутренности. Просто ее появление стало необходимостью, связанной с решением весьма прозаической задачи — повышения эффективности решения задач в системе пользователь — ЭВМ.

Говоря об эффективности компьютера, мы подразумеваем не только его возможности по решению каких-то определенных (например, вычислительных) задач, но и удобство работы с ним. Ведь программы решения задач не сваливаются с неба — их создают и с ними общаются пользователи. Поэтому качество процесса общения пользователя с компьютером в значительной степени характеризует возможности самого компьютера. Если «прогон» программы происходит без помощи пользователя, то решение задачи без него попросту невозможно: компьютер только выполняет программу решения задачи, а не решает ее. Процесс же решения происходит в системе пользователь — ЭВМ. И успех решения зависит от эффективности взаимодействия обеих сторон в этом диалоге.

Операционная система и способствует повышению эффективности такого взаимодействия пользователя и компьютера. Без операционной системы пользователю было бы худо. Компьютер же не пострадал бы совсем — его эффективность при прогоне готовой программы почти не изменилась бы. Но процесс создания и отладки программы пользователем был бы крайне затруднен.

ФУНКЦИИ ОПЕРАЦИОННОЙ СИСТЕМЫ

Прежде всего отметим, что всякая операционная система — это программная система, помогающая пользователю решать его задачи. Операционная система освобождает пользователя от многих рутинных и довольно нудных операций, связанных с использованием аппаратных средств компьютера — процессора, оперативной памяти, внешней памяти, печатающего устройства и т. д. Именно через операционную систему пользователь получает доступ к этим ресурсам, управляет работой компьютера, получает информацию, где находятся нужные ему программы и данные, узнает, куда направить полученные результаты.

А как же управляющее устройство?— возможно, воскликнет удивленный читатель.— Не его ли это функции? Почему же необходима еще одна специальная система, чтобы управлять работой компьютера?

Действительно, управляющее устройство — это аппаратные средства управления работой компьютера, связанные в основном с выполнением имеющейся программы. Это полностью автоматизированный процесс, быстрота выполнения которого определяет эффективность (точнее, производительность) компьютера, т. е. число операций в секунду, которое способен выполнить данный компьютер. Операционная же система является программным продолжением управляющего устройства. Именно она позволяет пользователю вмешиваться в процесс управления работой компьютера на стадии составления и отладки программы. А поводов для такого вмешательства у пользователя предостаточно.

Это, прежде всего, ответы на многочисленные вопросы, возникающие у пользователя: где во внешней памяти (например, на диске) найти программу, нужную пользователю, как ее приспособить для решения его задачи, как ввести исходные данные, куда направить готовую или подготавливаемую программу и т. д. и т. п. Таких вопросов и задач во время диалога с компьютером у пользователя возникает множество. И все они разрешаются при общении пользователя с операционной системой.

Таким образом, операционная система является набором программ, с помощью которых пользователю предоставляются все возможности компьютера, т. е. весь сервис, необходимый для эффективной работы пользователя. При этом важнейшей функцией операционной системы является организация и поддержание того, что называется...

ФАЙЛОВАЯ СИСТЕМА

Напомним, что *файлом* называют упорядоченный набор записей, выступающий как самостоятельный единый объект. В виде файлов хранятся программы, данные для их работы, любые тексты, таблицы, изображения и т. д. Каждый файл имеет свое имя, которое регистрируется в каталоге диска, на котором хранится этот файл. В каталоге имен файлов имеются сведения об их размерах и положении на диске.

Для манипулирования файлами пользователю следует знать несложный язык команд, с помощью которого он сможет активно вмешиваться в содержимое файлов. Такими командами в операционной системе CP/M, например, являются:

DIR — вывести каталог файлов диска,
REN — переименовать файл,
ERA — уничтожить файл,

TYPE — вывести файл на дисплей или принтер,
COPY — копировать файл.

Эти команды позволяют просмотреть на экране дисплея каталог файлов, записанных на выбранном диске, переименовать содержащиеся в нем файлы, скопировать их на другие носители (диски, магнитные ленты или на бумагу с помощью принтера), стереть его имя в каталоге и т. д. Естественно, что файловая система имеет и защиту от возможного искажения или стирания каких-либо очень важных файлов, например тех, где расположена операционная система (ведь операционная система, как любая программная система, хранится в виде файла во внешней памяти, обычно магнитном диске, и загружается в оперативную память для работы с компьютером).

Общение пользователя с внешними устройствами (дисплей, накопители на магнитных дисках, печатающее устройство, графопостроитель, каналы связи компьютера с другими ЭВМ и т. д.) осуществляется через специальные программы операционной системы, называемые *драйверами*. Так, если хотите вывести на экран дисплея содержимое определенного файла, достаточно указать номер диска и имя файла и, разумеется, команду его вывода на дисплей. В соответствии с этой командой файловая система по каталогу определит, где именно на диске расположен нужный файл, и сообщит эту информацию драйверу, который запустит нужный дисковод, переведет считывающие головки на нужную дорожку и считает файл в оперативную память. Далее вступит в работу драйвер дисплея, который переведет эту информацию на экран дисплея. Как видно, драйверы осуществляют взаимодействие компьютера с его внешними устройствами и отражают их специфику. При смене внешнего устройства надо менять и его драйвер в операционной системе.

Одной из самых популярных программ операционной системы является...

РЕДАКТОР

Вызвать программу редактора просто — достаточно набрать на экране слово ED (англ. editor — редактор). После этого можно производить любое изменение текста в том файле, с которым вы работаете. Это может быть программа, полученная вами с диска или составленная только что.

Процесс программирования в настояще время все больше сводится к редактированию текстов программ из многочисленных пакетов прикладных программ, имеющихся почти для всех компьютеров, особенно для персональных. Они удовлетворяют потребности практически всех пользователей. Оказывается, эти потребности не слишком разнообразны и отличаются лишь исходными

данными. Поэтому пользователю предоставляется богатый выбор прикладных программ, которые он использует, лишь незначительно редактируя их и вводя свои исходные данные. Именно это позволяет сделать программа редактора. Но редакторы бывают разные. Мы уже упоминали их при описании пакетов прикладных программ для персональных компьютеров. Теперь рассмотрим функции редакторов подробнее.

Текстовый редактор

Программой текстового редактора практически все время пользуются все, кто для написания книг, статей, писем и т. д. использует компьютер. Это прежде всего ученые, инженеры, журналисты, писатели, администраторы, клерки и т. д., т. е. люди, далекие от вычислительной техники (если не считать ученых и инженеров, использующих компьютер в своей профессиональной деятельности). Для этой категории «пишущих» пользователей нужна только программа редактора и файловая система для хранения написанного текста. Ведь программе редактора все равно, что редактировать,— программу на алгоритмическом языке, текст художественного произведения или научной статьи. Для него все это текст, в котором пользователь хочет сделать какие-то изменения. Зная несложный язык для общения с редактором, пользователь может легко изменять свой текст, как ему захочется, например убрать слово, предложение или абзац, вставить что-либо, заменить одно или несколько слов и предложений на другие и т. д.

Более того, редактор позволяет представлять текст на экране дисплея в том виде, в каком его хочет видеть пользователь. Например, определить размер страницы, выровнять поля, перенумеровать страницы текста, определить расположение графиков в тексте, выбрать шрифт. Для журналистов и писателей текстовый редактор позволяет обращаться к словарю синонимов и антонимов, что очень важно при стилистической обработке текста. Достаточно указать курсором заменяемое слово, вызвать список его синонимов из памяти компьютера (этот список появится в окне экрана дисплея) и указать курсом на выбранный синоним, синоним сразу встанет на место указанного слова. Сделает это программа текстового редактора.

Графический редактор

В помощь текстовому редактору выступает графический редактор, который предоставляет пользователю возможность построить нужный график. Для этого в памяти программы графического редактора хранятся несколько типичных видов гра-

фиков и диаграмм. Это, прежде всего, обычный график в виде кривой, точнее ломаной, выражающей зависимость одного фактора от другого. Пользователю следует ввести лишь значения координат точек графика и масштабы по осям, все остальное выполнит графический редактор. Есть диаграммы соотношений, с помощью которых удобно иллюстрировать рост или падение какого-то показателя во времени, что любят администраторы. Распределение какого-то ресурса удобно представлять в виде круговой диаграммы, сектора которой наглядно иллюстрируют долю этого ресурса, выделенного какому-то подразделению учреждения, и т. д. и т. п.

Весь этот сервис особенно удобен при оформлении отчетов и научных статей, что и делает программы редакторов, текстового и графического, очень популярными.

Выпускаются специализированные персональные компьютеры, снабженные т о л ь к о программами редакторов. Такой компьютер с накопителем на магнитном диске и принтером (печатающим устройством) обеспечивает идеальные условия и место для работы огромного числа пользователей.

Таковы некоторые внешние проявления операционной системы, обеспечивающей эффективное взаимодействие компьютера и пользователя. Но есть и другая, пожалуй более важная, функция операционной системы — распределение ресурсов компьютера во время его работы, куда доступ пользователю закрыт.

ОПЕРАЦИОННАЯ СИСТЕМА — РАСПРЕДЕЛИТЕЛЬ РЕСУРСОВ

Под ресурсом, вообще говоря, понимают всякое ограниченное средство, необходимое для удовлетворения потребностей системы. Ресурсом, например, является время (ограниченность его не вызывает сомнений) или материальные средства (например, в денежном выражении), также ограниченные.

В компьютере ресурсами являются процессорное время, оперативная память, устройства ввода-вывода и прикладные программы. На каждый из этих ресурсов могут претендовать как сами пользователи, так и процессы, порождаемые этими пользователями. Рассмотрим, как управляет операционная система каждым из этих ресурсов.

Распределение процессорного времени

Процессор является центральным органом каждого компьютера и самым его дорогим ресурсом. Поэтому загрузка процессора определяет эффективность работы компьютера — чем больше загружен процессор, тем выше производительность компьютера.

Любой простой процессора снижает быстродействие компьютера. Именно поэтому так важно умело распоряжаться ресурсом процессорного времени, особенно в мощных компьютерах.

Рассмотрим, как решается эта проблема при решении компьютером сразу нескольких задач. Эта ситуация типична при обработке мощным компьютером потока поступающих на него задач, например в автоматизированных системах управления производством (АСУП).

Характерной особенностью процесса решения почти любой задачи является то, что время от времени в ней возникает необходимость в получении информации из внешних запоминающих устройств (чаще всего с магнитных дисков). Для этого необходимо обратиться к операции ввода-вывода, на которую затрачивается время (по крайней мере равное времени одного оборота диска — примерно 0,02 с). Но за это время процессор производительностью в 1 млн операций в секунду может выполнить свыше 20 тыс. операций, т. е. решить небольшую задачу. Поэтому задачу, которая ждет получения необходимой информации, целесообразно временно снять с процессора и загрузить его выполнением другой задачи, готовой к обработке (из очереди). А снятая задача после получения информации снова ставится в очередь к процессору. Такой режим работы называют *многопрограммным* или *мультипрограммным*. Он обеспечивает высокую загрузку процессора: незначительные простои происходят только в моменты перезагрузки процессора — перехода с одной задачи на другую.

В многопрограммном режиме все решаемые задачи разделяются на два класса — готовые к обработке задачи, которые ждут своего решения на процессоре, и заблокированные, ожидающие завершения ввода-вывода для получения информации и продолжения их решения. При получении этой информации задача второго класса переходит в первый. Схема многопрограммной обработки показана на рис. 6. Здесь ресурс времени процессора используется наиболее производительно — прерывания его работы происходят только при завершении решения задачи или блокировке решения, если необходимо обращение к вводу-выводу, т. е. в моменты, предвидеть которые заранее невозможно. Если бы решалась только проблема минимизации времени простоя процессора, то описанный режим обеспечивал бы указанное требование. Но при этом нужно еще сделать наименьшим время пребывания решаемой задачи в системе, т. е. сократить не только время ее решения, но и время ожидания в очереди к процессору.

Описанный режим распределения ресурса процессорного времени создает неблагоприятные условия для решения коротких задач. Действительно, всякая короткая задача вправе рассчитывать на то, что она будет быстро решена, да и возникает она обычно оперативно и поэтому требует очень быстрого решения. В человеческом общежитии проблема решения коротких задач

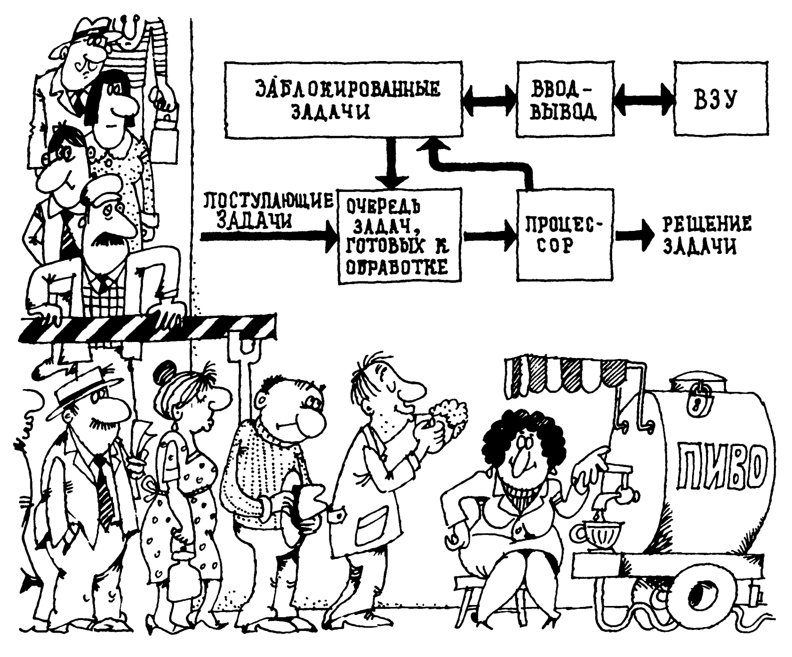


Рис. 6

решается просто — они идут без очереди. Вспомним реплики тех, кто идет без очереди: «Мне только спросить», или «У меня без сдачи», «Мне только коробок спичек», или «Мне только закрыть бюллетень» и т. д. Эти аргументы обычно принимаются, и очередь пропускает вперед «задачу», решение которой не займет много времени.

Но здесь возможны ошибки. По виду задачи трудно решить, какое именно время займет ее решение (как часто клиент, зашедший «на минуточку», задерживается надолго). Если заранее знать точно это время, то разумным правилом построения очереди к процессору был бы порядок по возрастанию времени их решения. При этом среднее время пребывания задачи в системе (оно складывается из времени ожидания в очереди и времени решения) было бы минимальным. Но именно этой информации нет при организации работы процессора.

Круговорот

Для преодоления этой трудности и был предложен режим квантования процессорного времени. В этом режиме каждой задаче при ее выходе на процессор выделяется квант времени Δ,

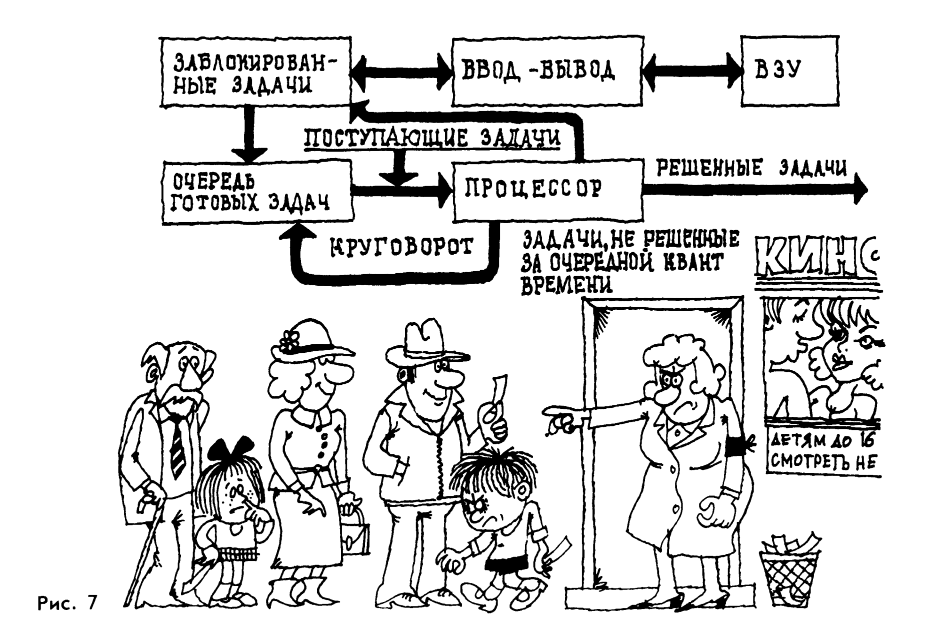


Рис. 7

равный, например, 0,01 с. Если задача за это время была решена, то процессор немедленно загружается другой готовой к выполнению задачей из очереди. Но решение этой задачи может быть приостановлено по двум причинам. Во-первых, исчерпан выделенный квант времени Δ, так как задача оказалась не очень короткой. И во-вторых, необходимость обращения к устройству ввода-вывода для получения информации с магнитного диска, требуемой для дальнейшего решения этой задачи. В первом случае незаконченная задача отправляется в конец очереди задач к процессору, который продолжит ее решение не позже чем через $n\Delta$ секунд, где n — число подготовленных задач, стоящих в очереди к процессору. Во втором случае (задача снимается из-за необходимости обращения к внешней памяти) процессор обращается к очередной задаче, а снятая задача дожидается необходимой информации и, получив ее, ставится в конец той же очереди готовых задач к процессору. Схема такой обработки (рис. 7) получила название круговорота (RR — Round Robin).

Действительно, большие задачи, не требующие ввода-вывода, как бы вращаются до тех пор, пока не будут решены. При этом новые задачи поступают на процессор без очереди, и их обработка начинается не позже чем через квант времени после их поступления. В результате короткие задачи (требующие для своего решения не больше одного кванта процессорного времени) будут решаться сразу. Если нужно два кванта, то задача будет пребы-

вать в системе время, равное $(n+1)\Delta$, где n — длина очереди и т. д.

Как видно, такая организация процесса вычислений позволяет быстрее решать те задачи, которые требуют меньшего времени, соблюден принцип скорейшего обслуживания того, кто требует меньших средств (за счет увеличения времени обработки длинных задач, разумеется). В этом случае среднее время пребывания задачи в компьютерной системе будет минимальным. При круговороте все задачи с одинаковой скоростью движутся к своему концу (решению). Поэтому чем короче задача, тем быстрее она будет решена и выйдет из системы.

— Round robin, — задумчиво повторил Мегрэ. — Где-то я уже слышал эти слова. Ведь в переводе с английского это означает «крутящаяся малиновка» или «крутящийся Робин». Позвольте, а не Робин Гуд ли это — легендарный стрелок и защитник справедливости?! Если это так, то смысл здесь не во вращении, а совсем в другом — в справедливости. Ведь, по преданию, именно Робин Гуд ввел в практику рыцарских турниров круговую систему — каждый с каждым. Эта система в отличие от олимпийской (с выбыванием) была справедливей: давала возможность каждому участнику турнира встретиться с каждым другим. Так сейчас проводят шахматные, футбольные и другие соревнования. Нетерпеливые олимпийцы придумали свою систему для того, чтобы быстрее из большого числа претендентов на лавровый венок венчать одного. При этом самый достойный легко мог отсеяться в результате неудачной жеребьевки, которая сведет его с сильным соперником не в конце, а в начале турнира. Этого не может произойти в круговой системе — в ней все равноправны! Так что вполне возможно Round robin здесь переводить не как «круговорот», а как предоставление равных возможностей.

Но и режим круговорота часто не устраивает пользователей. Коротких задач всегда значительно больше, чем длинных, и требуют они максимально быстрой обработки, так как обычно порождаются оперативно, например в режиме отладки программы пользователей. Это значит, что следует оказывать максимальное предпочтение коротким задачам за счет длинных, разумеется, которые, как правило, не требуют слишком уж быстрого решения. Здесь помогает введение динамических приоритетов.

Делается это следующим образом. Приоритет задачи увеличивается на единицу после очередной ее обработки на процессоре, а очередь образуется к процессору в порядке возрастания приоритетов. Таким образом, новые задачи имеют нулевой приоритет и решаются сразу по их появлении (здесь преимущества имеют задачи с низким приоритетом). Затем решаются задачи с единичными приоритетами, они однажды уже были обслужены процессором, и их решение было приостановлено или по окончании выделенного кванта процессорного времени, или из-за необ-

ходимости получить информацию из внешней памяти. Далее на процессор поступают задачи со вторым приоритетом и т. д. Причем при наличии нескольких задач с одним и тем же приоритетом они решаются в порядке их поступления в очередь данного приоритета.

Легко заметить, что задача с высоким приоритетом (она трудоемкая и уже много раз проходила через процессор) будет выходить на процессор только тогда, когда нет задач с более низким приоритетом. Такие задачи называют фоновыми — обычно они решаются, когда процессор свободен от решения срочных оперативных задач. В описанном режиме круговорота с динамическими приоритетами не надо делить задачи на срочные и несрочные. Это происходит автоматически, если срочные задачи короткие, что, как правило, и бывает на самом деле.

Если вы вышли на такой компьютер с большой задачей, то не возмущайтесь, что она решается дольше, чем следовало бы по вашим расчетам. Просто она стала фоновой и ждет свободных квантов процессорного времени. Но зато в режиме отладки, когда нужно быстро опробовать различные варианты программ и подпрограмм, такой компьютер будет очень оперативно пропускать все ваши задачи.

Все эти функции — назначение приоритетов, соблюдение очередей и т. д. — выполняет операционная система большего компьютера. Заметим, что таких проблем не стоит перед операционной системой персонального компьютера, ведь он персональный и решает только одну задачу своего пользователя.

Распределение ресурса памяти

Другой важнейший ресурс любого компьютера — оперативная память, которая, как известно, ограничена, что и создает трудности при ее распределении. Выходом из этого положения является подключение внешней памяти, которая значительно больше, но значительно медленней оперативной. Это старый прием: не найдя нужной информации в своей оперативной памяти (мозгу), мы обращаемся к внешней памяти — справочникам, монографиям, специалистам и т. д. Очевидно, что время обращений к такой памяти значительно больше, чем к своей, но и сведений она содержит значительно больше.

Чтобы преодолеть указанные трудности, и был предложен остроумный механизм *виртуальной памяти* — памяти, кажущейся пользователю оперативной, но не являющейся ею в действительности. Реализует такую память операционная система.

Для этого информацию, которую надо хранить в памяти (программы и данные), сегментируют в виде небольших страниц в 1024 — 2048 слов (машинных, разумеется). Каждая такая страница имеет свой номер и перечень того, что в ней записано. Теперь нужно решить, какие из них хранить в оперативной

памяти, а какие во внешней. Для этого все страницы разбиваются (сначала случайно) на две неравные части. Меньшая помещается в оперативную память (быструю, но малую), а остальное — во внешнюю (медленную, но большую), чаще всего на магнитные диски.

Процессор может работать лишь с теми страницами, которые расположены в оперативной памяти. Если же ему понадобится страница, которой там нет, то операционная система найдет ее на диске и переведет в оперативную память с помощью операции ввода-вывода, на которую придется затратить довольно много времени, что, естественно, задержит выполнение программы. Поэтому ситуацию, когда нужно переводить страницу с диска в оперативную память, называют страничным сбоем, подчеркивая нежелательность этой ситуации. При этом одну страницу из оперативной памяти следует вернуть во внешнюю память, чтобы заменить ее новой. Какую именно страницу следует замещать, решает алгоритм замещения страниц (точнее, программа), реализуемый операционной системой. Этот способ получил очень образное название динамической подкачки страниц. Действительно, здесь страницы как бы «перекачиваются» из внешней памяти в оперативную и обратно.

В качестве таких алгоритмов могут выступать разные правила, например замещение случайной страницы или самой старой страницы, когда замещается страница, дольше всех находившаяся в оперативной памяти, или страницы, которая дольше всех не использовалась процессором, и т. д. Так или иначе, но алгоритм замещения должен удалять из оперативной памяти те страницы, которые в дальнейшем не потребуются для работы процессора.

Как видно, алгоритм замещения страниц этим пытается заглянуть в будущее. Дело это, как известно, крайне рискованное. Но, если будущее хоть немного похоже на настоящее, то можно пытаться его прогнозировать. Это и делает алгоритм замещения страниц, реализованный в операционной системе. В результате работы такого алгоритма в оперативной памяти окажутся те страницы памяти, которые чаще нужны для решения текущих задач компьютера.

Значит ли это, что любой алгоритм замещения страниц всегда хорош? Разумеется, нет! В одних случаях лучше один, а в других — другой. А лучшим будет тот, который обеспечивает минимальное среднее число страничных сбоев в единицу времени, что и дает максимальную производительность компьютера. Страничные сбои неизбежны, но число их должно быть минимальным. Это и является критерием эффективности алгоритма замещения страниц.

Трудности выбора наилучшего алгоритма замещения страниц настолько велики, что иногда заставляют вообще отказаться от описанного механизма замещения. В этом случае в оперативной

памяти хранится лишь каталог страниц, где указано, что именно находится на каждой странице, а сами страницы — во внешней памяти. Каждое обращение к памяти вызывает при этом неизбежный страничный сбой. Но это не снижает производительности компьютера, если он работает в многопрограммном режиме: его процессор тут же будет загружен следующей задачей, готовой к решению.

Как видно, при виртуальной памяти пользователь не ограничен емкостью оперативной памяти. В его распоряжении оказывается огромное поле виртуальной памяти, с которым он и работает, считая ее оперативной. Разница лишь в том, что быстродействие компьютера с виртуальной памятью несколько ниже — неизбежные страничные сбои снижают производительность процессора (в однопрограммном режиме, разумеется) из-за простоев при ожидании ввода-вывода при сбое.

И еще многое другое ...

... возложено на операционную систему. Так, в компьютерных системах коллективного пользования операционная система имеет программу *супервизора* (англ. supervisor — сверхнаблюдатель). В задачу его входит контроль за состоянием и работой всех устройств компьютера и наблюдение за вычислительным процессом, чтобы организовать эффективную совместную работу всех действующих программ, устанавливая привилегии одним и иногда «наказывая» других. Супервизор обеспечивает синхронизацию между процессами выполнения программ и устройствами компьютера, осуществляет распределение всех ресурсов и услуг компьютерной системы между ее пользователями и многое другое.

Другой важной функцией операционной системы является *тестирование* отдельных узлов компьютера. Если у пользователя возникает подозрение в ошибочной работе каких-то узлов, ему достаточно обратиться к программам тестирования этих узлов, которыми располагает операционная система, что она немедленно выполнит. При этом на экране дисплея будет указано, какой именно блок компьютера вышел из строя и требует замены.

— Ну здесь все ясно! — бодро сказал Мегрэ. — Операционная система — это администратор компьютера, распределяющий его ресурсы так, чтобы пользователю было хорошо. Это хороший администратор! Не чета нашим чинушам в управлении криминальной полиции, которых волнует не дело, а политические интриги и собственная карьера.

— Не торопитесь, шеф, — заметил Поль. — К сожалению, устройство компьютера отражает иерархическую организацию управления всякой сложной системой. Эта иерархия точно такая же, как в нашем криминальном управлении: министр, его замы, начальники полиции и префектур, комиссары и, наконец,

на самом основании полицейские, которые осуществляют функцию соблюдения законов. Все, что выше них, — это администраторы, которые должны обеспечивать эффективную работу полицейских.

— Постойте! — перебил его Мегрэ. — Вы что и меня зачислили в администраторы?

— Конечно! Каждый из нас выступает в двух ипостасях — исполнитель по отношению к более высокому уровню иерархии и администратор — к более низкому. И в этом нет ничего плохого. Такова специфика всякой иерархической системы, которая доказала свою эффективность за миллионы лет существования. Ведь не мы ее придумали, а Природа ...

— Не хотите ли вы сказать, — перебил его Мегрэ, — что иерархия нашего общества, а вместе с ним и компьютера заимствованы у Природы? Где она там?

— Да хотя бы в организации любой популяции, в обезьяньем стаде например. Ведь там существуют очень строгие и стойкие иерархические отношения между животными, которые редко изменяются. Проявляются они, например, в порядке кормления: сначала вожак, потом его «приближенные» и т. д.

— Так в чем же преимущество такой иерархии? Неужели потребность быстрее «набить брюхо» оказалась столь существенной, что было закреплено Природой в специальной структуре организации биологических сообществ? Да еще эффективно используется в компьютере?

— Да! Причина этого — дефицит ресурсов. Если бы дефицита не было, то и преимущества иерархии свелись бы к минимуму. Но при наличии дефицита должен существовать орган, распределяющий ресурсы. Именно это и образует иерархические отношения подчинения.

— Но какое это имеет отношение к компьютеру? — раздраженно заметил Мегрэ.

— Самое непосредственное! Ведь при работе современного компьютера взаимодействует множество программ, которые требуют для своего выполнения определенных ресурсов в виде процессорного времени, оперативной памяти, устройств ввода-вывода, разнообразных внешних устройств ... Эти ресурсы всегда ограничены, и для эффективной, производительной работы компьютера их просто не хватает. Вот и приходится создавать орган, ведающий распределением ресурсов. Это и есть одна из основных функций операционной системы.

— А если программа была бы одна?

— То и не нужно было бы операционной системы. Так и было — первые компьютеры работали по одной программе и ей предоставлялись последовательно все необходимые ресурсы. Никаких ресурсных конфликтов возникнуть в принципе не могло: некому было конфликтовать. Поэтому не было надобности в операционной системе. Появилась она в компьютерах второго поколения, когда возникла необходимость в многопрограммном режиме.

— Да, но когда я работаю на персональном компьютере с одной своей программой, конфликтам вроде бы возникнуть неоткуда? — спросил Мегрэ. — Тогда зачем операционная система?

— В современном компьютере (в том числе и в персональном) имеется большое число программ, которые необходимы для простого и эффективного решения вашей задачи. Без них вам пришлось бы программировать каждый шаг

работы компьютера. Например, для вызова своего файла с дискетки вам достаточно назвать его имя и оператор вызова. При этом включаются уже имеющиеся программы поиска этого файла на дискетке, его считывания и передачи на экран дисплея. Каждая из этих программ требует своих ресурсов, распределением которых и ведает операционная система. Таких вспомогательных программ в компьютере множество. Именно они и создают многопрограммный режим работы компьютера и ... необходимость в иерархии управления, т. е. в операционной системе.

— Эдак вы, дружище, оправдаете любой бюрократический аппарат, — сварливо заметил Мегрэ.

— Да нет, — улыбнулся Поль. — Это лишь выявляет причины появления бюрократии. Они объективны. Но возникнув, всякая бюрократия начинает обеспечивать свое благополучие даже в ущерб всей системе управления.

— Какой же ущерб наносит операционная система? — язвительно спросил Мегрэ.

— Она сама съедает ресурсы компьютера: ведь операционная система — большая программа и поэтому требует значительных затрат процессорного времени за счет времени решения задач пользователя, большой оперативной памяти за счет того же бедного пользователя. Именно это и породило грустную шутку пользователей: «Что такое слон? — Это мышь с операционной системой». Так что иногда приходится отказываться от услуг наиболее совершенных и поэтому громоздких операционных систем, заменяя их простыми, но более экономными.

— Я вижу, что в компьютере, — заметил Мегрэ, — те же проблемы, что и в жизни.

— Конечно, ведь компьютеры по своей сложности приближаются к нам и, следовательно, неизбежно обретают наши недостатки — недостатки сложных систем, — грустно закончил Поль.

5. ЯЗЫК МОЙ — ДРУГ МОЙ
(Языки общения с компьютером)

Всем известны слова великого М. Ломоносова: «Карл V, римский император, говаривал, что испанским с богом, французским с друзьями, немецким с неприятелем, итальянским с женским полом говорить прилично». Но нас они интересуют не как похвала универсальности русского языка, на котором «со всеми оными говорить пристойно», а как признание того, что для разных целей следует применять различные языки. Наверняка можно с женщинами объясниться и на немецком (ведь немцы так и поступают), но лучше ... на итальянском — таково мнение знатоков.

Точно так же и при общении человека с компьютером в процессе решения им конкретных задач. Нет языка, наилучшего для описания и решения всех задач с помощью компьютера. Для каждой задачи всегда один язык будет лучшим по выбранному критерию (краткости, ясности, простоте и т. д.). Этим и объяс-

няется обилие и многообразие языков общения с ЭВМ — каждый из них создавался для решения определенных задач, определенного типа компьютера, а также определенного уровня знаний и опыта пользователя в программировании. Поэтому, общаясь с ЭВМ на каком-либо языке и испытывая при этом неудобства, не следует сетовать на язык — просто он создан для других задач, компьютеров, пользователей. И вам следует либо подобрать язык с учетом специфики своей задачи и компьютера, либо ... создать свой язык. Эту последнюю возможность используют крайне редко, когда на это есть веские основания. Именно так появляются новые языки программирования.

Мы уже познакомились с общими идеями построения языков общения пользователя с компьютером (см. гл. 3). Выяснили, что языки бывают низкого (машинный и символический языки) и высокого уровня. Последние представляют наибольший интерес для широких кругов пользователей: они максимально приближены к естественному языку человека, что очень важно для освоения техники программирования. Языки высокого уровня не требуют от пользователей знаний об устройстве и функционировании компьютера, на котором будет решаться задача. Языки бывают *универсальные* и *специализированные*. Последние ориентированы на узкий класс задач и строятся обычно на базе универсальных. Поэтому рассмотрим лишь основные универсальные языки.

Сразу следует отметить, что все эти языки пользуются английским алфавитом и языком для изображения операторов, переменных, меток. Причина этого вовсе не в каких-то удобствах английского языка, а в том, что сами языки высокого уровня и первые программы для компьютеров появились в англоязычных странах. И хотя есть неплохие русские варианты языков высокого уровня, программисты, а вслед за ними и пользователи обычно предпочитают английский. При этом программы понятны всему миру, с чем, естественно, нельзя не считаться. А если необходимо ввести русские слова, их пишут латинскими буквами. Так, в программах появляются довольно забавные образования типа VREMJA или DEN NEDELI. В этом ничего страшного нет, хотя учителя русского языка наверняка будут шокированы.

ПРОГРАММИРОВАТЬ ИЛИ МОДЕЛИРОВАТЬ?

Все языки общения с компьютером подразделяются на два больших класса языков программирования и моделирования. В первом случае используется в основном вычислительная функция компьютера. Языков программирования много хотя бы потому, что считать на компьютере стали раньше, чем реализовывать его невычислительные функции. Но компьютеры используются не

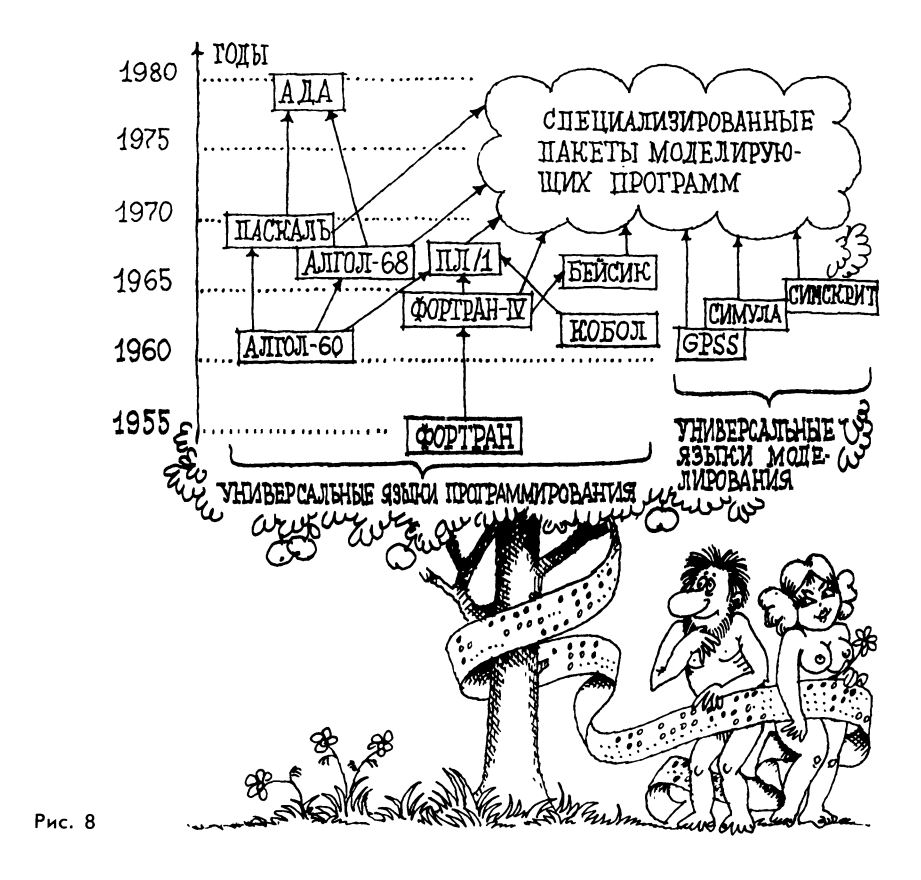


Рис. 8

только для вычислительных задач. Более того, невычислительных функций, выполняемых компьютером, сейчас значительно больше. Это и обработка текста (например, при компьютерном переводе с одного языка на другой), и хранение больших массивов информации, и выдача требуемой по запросу пользователя и т. д. Но одной из важнейших невычислительных функций является моделирование. Если выполнение невычислительных функций занимает 90 % общего мирового компьютерного времени, то моделирование — добрую половину от этих 90 %.

Сам по себе процесс компьютерного моделирования связан с воспроизведением (воссозданием, имитацией) поведения интересующего нас объекта, процесса, явления, системы. Значительно экономичней, быстрей, точней и т. д. можно определить свойства и характеристики объекта, не создавая его реально, а на его модели. Моделирование отвечает на вопрос: а что будет, если...? Таких вопросов задается много как на стадии проектирования, так и эксплуатации различных систем.

Программа такого моделирования может быть составлена на любом языке программирования. Но ввиду специфики процесса моделирования и его большой распространенности лучше для этого создать специальные языки. Их так и называют *языками моделирования*. Эти языки существенно отличаются от языков программирования своей узкой специализацией в направлении моделирования и обработки получаемой информации.

Языки обычно являются развитием других, более простых языков. Такое «дерево языков», преемственность различных языков программирования, показано на рис. 8.

Начнем с *языков программирования*. Их много, и отличаются они тем, что обслуживают определенный, достаточно широкий класс преимущественно вычислительных задач.

ФОРТРАН

Первым и самым распространенным по сей день является язык Фортран (сокращение слов ФОРмульный ТРАНслятор, т. е. формульный переводчик). Он был создан в 1956 г. в ответ на практические потребности загрузки мощных компьютеров, появившихся к этому времени. Они стали простаивать из-за недостатка программ, так как программисты не справлялись с их подготовкой. Тогда решили подключить к программированию самих пользователей ЭВМ. Вот так и появился Фортран — первый язык программирования, ориентированный прежде всего на пользователей.

За более чем 30-летнюю историю Фортрана на этом языке написано такое количество нужных и важных программ, что отказаться от него или заменить каким-то новым языком не только нецелесообразно, но просто расточительно: будет утрачен огромный фонд программ, написанных на этом языке. Именно поэтому Фортран иногда называют вечным языком — он входит обязательно во все существующие серьезные программные системы и наверняка будет входить в будущие. Более половины пользователей предпочитают Фортран другим языкам. Именно они придумали лозунг: «Настоящие программисты пишут только на Фортране». Кроме шутки, здесь есть доля истины. Фортран — не застывший язык, он совершенствуется, последующие версии (Фортран-II, Фортран-IV, Фортран-77) включают все предыдущие и предоставляют дополнительные возможности.

В Фортран-программе операторы не нумеруются и выполняются в порядке их следования в программе. При необходимости изменения этого порядка используются так называемые метки — целые числа, поставленные перед соответствующими операторами. Операция присвоения в Фортране реализуется в виде равенства, например S=S+X означает, что переменной S присваивается новое значение, равное сумме предыдущего значения S и значения X. Оператор цикла здесь имеет вид: DO 5 M=1,N, т. е. делать

(DO) цикл до метки 5 при M, изменяющемся от 1 до N (всего N раз), где N — заданное число. Например, при вычислении среднего из N чисел $s=\frac{1}{N}(x_1+\ldots+x_N)$ удобно воспользоваться этим оператором. Для этого введем сумму m чисел:

$$s_m=x_1+\ldots+x_m,$$

тогда $s_m=s_{m-1}+x_m$ и $s=s_N/N$. В результате получаем такую программу:

S=0	Присвоение нуля начальному значению s, чтобы стереть число, которое могло остаться в ячейке S от прошлой работы компьютера.
DO 5 M=1, N	N-кратный цикл вычисления до оператора с меткой 5, т. е. следует N раз выполнять вычисления от этого оператора (цикла) до оператора с меткой 5.
5 S=S+X(M)	Вычисление $s_m=s_{m-1}+x_m$. X(M) соответствует x_m, а 5—метка.
S=S/N	Деление полученной суммы s_N на N.
TYPE*S	Вывод значения s в свободном формате (это обозначено звездочкой *), т. е. значения, которое получилось (без округления), и в любом месте экрана дисплея.

Для работы этой программы следует предварительно ввести или иметь в памяти компьютера значения чисел $x_1,\ldots,x_N$, т. е. массив X(M).

Всем хорош Фортран, но все-таки сложен для многих пользователей, которым желательна прежде всего простота. И такой язык был изобретен на базе Фортрана. Это Бейсик (BASIC — многоцелевой язык символических инструкций для начинающих).

БЕЙСИК

Сейчас, пожалуй, это самый популярный язык программирования и общения с ЭВМ среди начинающих пользователей. Главным его достоинством — ради этого он и был создан — является простота (его иногда в шутку называют языком для простаков). Научиться основным приемам работы с Бейсиком можно за час-полтора, именно поэтому его рекомендуют для начального обучения программированию. Но это вовсе не означает, что Бейсик не позволяет составлять программы для решения сложных задач. Просто они скорее всего будут несколько хуже, чем программы, составленные на «больших» языках. А вот простые задачи с помощью Бейсика решаются лучше: другие языки всегда обладают избыточностью, которая усложняет решение простых задач.

Сразу отметим, что Бейсик имеет несколько «диалектов» версий и единого стандарта на него нет. Есть просто Бейсик, Бейсик-

плюс, расширенный Бейсик и множество других Бейсиков. Как правило, каждый новый тип компьютера снабжается своим Бейсиком, отличающимся от других. Это нужно всегда иметь в виду. Знание одного из диалектов Бейсика не гарантирует успеха в программировании на другом диалекте. Это значит, что, зная Бейсик и переходя на другой компьютер, следует всегда поинтересоваться особенностями Бейсика на этой машине.

Так уж получилось, что вопреки общему правилу машинной независимости языков высокого уровня Бейсик оказался машинно-зависимым исключением. Трудно сказать, почему так получилось. Наверное, потому, что к Бейсику разработчики компьютеров не относились серьезно и «перекраивали» его под свои концепции. С Фортраном такого произойти не могло: не удалось бы воспользоваться уже накопленными программами. На Бейсике же таких программ было немного. Именно поэтому у всех Бейсиков нет единой характерной черты, кроме простоты да возможности диалога с компьютером в процессе программирования. Такой диалог очень облегчает программирование и обеспечивается режимом интерпретации, характерным для Бейсика. В любой момент вы можете запустить написанную программу с помощью оператора RUN (прогон). Этим оператором начинается выполнение программы в режиме интерпретации, а прекращается, как только встречается оператор STOP или END (конец).

Все операторы Бейсик-программы обычно нумеруются через 10, чтобы можно было делать вставки. Номера операторов играют в Бейсике роль меток. Для записи программы используются лишь прописные латинские буквы, как в Фортране.

Основные операторы Бейсика: LET (пусть) — оператор присваивания (хотя в некоторых версиях Бейсика он опускается); PRINT (печать) — оператор вывода результата, который выведет на печать то, что поставлено в кавычки после этого оператора и вставит полученные значения; ↑ — возведение в степень; ✱ — произведение; / — деление. Например, вычисление выражения $(1{,}23^{3{,}5}\sin\frac{8{,}35}{3{,}7})/17{,}6$ осуществляет следующая программа:

```
10  LET X=(1.23↑3.5)✱SIN(8.35/3.7)/17.6
20  PRINT «X=», X
30  END
    RUN
```

при выполнении которой получим на экране дисплея (или на бумаге печатающего устройства) результат в виде X=0.090738. Например, Бейсик-программа вычисления среднего из N чисел имеет вид

```
10 S=0
20 FOR K=1 TO N          Заголовок цикла по K от 1 до N
30 S=S+X(K)
40 NEXT K                Переход к следующему (next — следующий)
                         значению K (т. е. K→K+1) и возвращение
                         к строке 30, как к началу цикла.
50 S=S/N
60 PRINT S               Печать результата
70 STOP
   RUN
```

В результате работы программы на экране дисплея будет выведено полученное значение S.

Почти все современные ЭВМ, особенно персональные, снабжены транслятором с языка Бейсик, работающим в режиме интерпретации (см. гл. 4). Это значит, что пользователь может выполнить сразу написанную на Бейсике часть программы.

Но не следует думать, что Бейсик — «игрушечный» язык. С его помощью можно решать достаточно сложные задачи. А для очень сложных имеется так называемый расширенный Бейсик, имеющий, например, матричные операции, которые позволяют с помощью одного оператора преобразовывать большие таблицы (матрицы). Есть версии Бейсика, работающие в режиме компиляции, что значительно ускоряет прогон программ, написанных на этом языке.

Фортран, Бейсик, Кобол, ПЛ/1 (см. рис. 8) являются фирменными языками, т. е. языками, разработанными фирмами-изготовителями ЭВМ для эффективного использования их продукции. Но одновременно с этим языки общения человека и ЭВМ были и до сих пор являются объектом чисто научного изучения, результатом которого стала разработка международной группой ученых научно обоснованных языков программирования. И первым из них был Алгол.

АЛГОЛ

Создан в 1960 г. (его поэтому иногда называют Алгол-60) и оказал большое влияние на развитие языков программирования (см. рис. 8).

Алгол-программа всегда начинается словом begin (начало) и кончается словом end (конец), их поэтому называют операторными скобками, записывается и прописными и строчными латинскими буквами, причем в публикациях операторы записываются полужирным шрифтом. Например, оператор условного перехода **if...then...else** (если..., то..., иначе...), где вместо многоточий стоят различные выражения, позволяет делать многое. Например, строка программы

$$\text{tf } A > 0 \textbf{ then } X := A/B$$

означает, что при $A > 0$ переменной X следует присвоить ($:=$) значение отношения A/B, а при $A \leqslant 0$ перейти к следующему оператору в программе. А строка

$$\textbf{if } X \geqslant C \textbf{ then go to } M \textbf{ else } R := 1$$

означает, что при $X \geqslant C$ следует переходить **(go to)** к оператору с меткой M, а иначе (англ. else — иначе). т. е. при $X < C$, переменную R приравнять к единице и перейти к следующему по порядку оператору.

Например, Алгол-программа вычисления среднего из *N* чисел:

	begin	Начало программы
	K:=1; S:=0	Задание начальных значений переменных K и S
2	**if** K=N **then go to** 1 **else** S:=S+X(K)	Если K=N, то перейти к оператору с меткой 1, если иначе (при K<N), то значение S увеличить на X(K) и перейти к следующему оператору
	K:=K+1; **go to** 2	Увеличить K на единицу и перейти к оператору с меткой 2
1	S:=S/N	Определить среднее значение, 1 — метка
	write S	Записать результат
	end	Конец

Применяют Алгол в научно-технических расчетах и научно-исследовательских работах. Развитием Алгола-60 является Алгол-68, созданный в 1968 г., который предоставляет пользователю бóльшие возможности.

ПАСКАЛЬ

Этот язык — прямое развитие направления Алгола (см. рис. 8), создан в 1969 г. и стал очень популярен в настоящее время. Причиной этого является прежде всего его простота. Так, описание языка Паскаль занимает всего 30 страниц текста. Именно поэтому его часто используют для обучения приемам программирования. Транслятор с Паскаля также прост и занимает мало места в памяти, что особенно важно для мини- и микроЭВМ, имеющих оперативную память малой емкости. И, наконец, в Паскале есть достаточно сильные средства для написания так называемых системных программ, для чего обычно используют язык ассемблера. Такая универсальность и компактность Паскаля сделала его наряду с Бейсиком очень популярным, особенно для персональных компьютеров. Например, Паскаль-программа вычисления среднего из *N* чисел:

BEGIN	Начало программы
K:=1; S:=0	Задание начальных значений
WHILE K<N DO BEGIN S:=S+X(K); K:=K+1	Пока (WHILE) K<N, делать (DO) $s_k = s_{k-1} + x_k$ и K→K+1, а при K=N обращаться к следующему оператору
END	Конец цикла (его начало было обозначено DO BEGIN)
WRITE (среднее значение S/N)	Писать слова «среднее значение», за которыми следует значение среднего (S/N) — это вывод результата
END	Конец

В результате работы этой программы получим следующий текст на экране дисплея:

среднее значение ... ,

где вместо точек будет стоять число — полученное среднее значение.

Сравнивая приведенные программы для вычисления среднего значения, написанные на разных языках (Фортране, Бейсике, Алголе и Паскале), легко заметить, что они примерно одинаковы. И ни один из рассмотренных языков не обладает заметным преимуществом перед другими при решении этой задачи, да и других вычислительных задач. Преимущество проявляется тогда, когда придется обратиться к библиотеке программ, чтобы найти и использовать уже готовую программу для своих целей. И лучше будет тот язык, который обеспечит решение этой задачи за минимальное число обращений к памяти и не потребует высокой квалификации пользователя.

Как видно, все зависит от задачи, пользователя, организации библиотеки программ, самого компьютера, его внешней памяти и многих других обстоятельств. Так что нельзя заранее предугадать, какой язык лучше, — все зависит от обстоятельств. Это и гарантирует каждому языку широкую область применения.

АДА

Этот язык создан в 1979 г., является следующим продолжением направления Алгола в программировании. Его основное назначение — программирование работы самых разнообразных систем управления на ЭВМ и вообще сложных программных систем. Известно, что создание больших программных систем затрудняется тем, что они ненадежны ввиду неизбежных ошибок, допускаемых при программировании. В Аде большое внимание

уделяется обеспечению надежности программ даже в ущерб легкости их написания. Поэтому основным элементом программы на языке Ада являются подпрограммы. Подпрограммы есть и в других языках, но там они играют вспомогательную, а здесь главную роль. Каждая подпрограмма состоит из двух частей: описательной и расчетной. В описательной части определяются очень подробно все логические понятия, используемые в данной подпрограмме: исходные данные, их тип, переменные, их тип, имена, возможные значения переменных и т. д. Расчетная часть подпрограммы представляет собой последовательность операторов, с помощью которых описываются действия, выполняемые подпрограммой.

Таким образом, каждая подпрограмма вполне самостоятельна и фактически является программой. Это позволяет параллельно писать и отлаживать сразу несколько подпрограмм, что очень важно при создании больших программных комплексов одновременно многими программистами. Такие комплексы необходимы для управления сложными технологическими процессами, летательными аппаратами, сложным научным экспериментом и т. д. Именно эти свойства языка Ада привлекают к нему внимание создателей больших и сложных систем управления на базе ЭВМ.

КОБОЛ

Язык Кобол (COBOL — общий язык, ориентированный на экономические применения) создан в 1960 г. для применения в экономике, точнее для обработки деловой информации. Кобол-программа выглядит как ряд предложений из английских слов, т. е. имеет вид текста на естественном языке, что значительно облегчает овладение этим языком пользователями — экономистами, администраторами и др. Характерной чертой его является то, что в качестве переменной может быть сложное сочетание букв, цифр и значков, например целая анкета, накладная или счет, необходимые преобразования которых и позволяют выполнять операторы Кобола.

ПЛ/1

Этот язык является попыткой совместить все лучшее, что есть в Алголе и Коболе. Главная цель его создания — иметь в языке средства, необходимые всем категориям программистов. Это очень мощный язык, но и очень громоздкий (поэтому его иногда не без сарказма называют «монстром»). ПЛ/1 может эффективно обслуживать программирование как экономических, так и научно-технических задач. Характерной его чертой являются многочисленные правила умолчания, которые позволяют компьютеру (точнее, его транслятору) самому определять то, что не

определено пользователем в программе. Например, используя букву N для обозначения целого числа, не нужно его специально описывать как целое (integer), как это требуется, например, в Алголе. Это сделает сам транслятор по правилу умолчания. Но для того, чтобы N могло быть десятичным числом, нужно его описать как вещественное (real). Правила умолчания помогают программировать, но затрудняют жизнь начинающим программистам: их надо знать и все время помнить.

GPSS — УНИВЕРСАЛЬНЫЙ ЯЗЫК МОДЕЛИРОВАНИЯ

Самым ярким представителем языков моделирования является язык GPSS, что в переводе с английского означает «общецелевая система моделирования». Этот язык высокого уровня позволяет моделировать поведение вероятностных систем, например систем массового обслуживания, вычислительных. Само по себе моделирование производится путем изменения состояния моделируемого объекта и сводится к генерации этих изменений, обычно случайных. Например, моделируя процесс обслуживания в магазине, достаточно генерировать случайные времена появления покупателей и окончание их обслуживания.

В соответствии с этим язык GPSS имеет операторы: GENERATE — генерировать и ADVANCE — задерживать (например, на время обслуживания). Числа, которые следуют за этими операторами, характеризуют свойства случайных процессов, которые моделируются. Например, GENERATE 20, 10, 10, 25, 2 означает, что надо генерировать поток клиентов, приходящих через случайные интервалы времени (20 ± 10), например, секунд (т. е. интервал времени между появлением двух клиентов случаен и распределен в пределах от 10 до 30 с), причем первый клиент приходит в случайный момент в интервале от 0 до 20 с от начала работы ($0 = 10 - 10$; $20 = 10 + 10$); всего должно быть смоделировано появление 25 клиентов, имеющих два приоритета — первый, идущих без очереди, и второй. Процесс обслуживания характеризуется временем пребывания клиента на месте обслуживания и моделируется оператором задержки. Например, ADVANCE 12, 8 означает, что на обслуживание клиента требуется случайное время, распределенное в интервале 12 ± 8 с, т. е. от 4 до 20 с.

В результате такого моделирования можно определить такие важные характеристики моделируемой системы, как, например, ее пропускную способность, длину очередей, среднее время пребывания клиентов в системе и многое, что очень важно знать на стадии проектирования или анализа вероятностной системы. При этом обычно все интересующие характеристики процесса обслуживания вычисляются автоматически и остается лишь запросить нужные вам показатели моделируемого процесса.

Существующие универсальные языки моделирования GPSS, Симула, Симскрипт и другие в сочетании с универсальными языками программирования позволяют создавать специализированные пакеты моделирующих программ (см. рис. 8) для решения таких важных практических задач, как моделирование и управление дискретным производственным процессом, процессом обработки информации вычислительной системы, транспортными потоками и др. Эти пакеты сочетают в себе достоинства языков программирования и моделирования, но в более узких специальных областях.

ЯЗЫКИ ОПЕРАЦИОННОЙ СИСТЕМЫ

Операционная система (ОС), на которую возлагаются функции координации работы компьютера, выступает в качестве посредника при общении человека и ЭВМ. Она имеет собственный язык — язык команд (его называют обычно языком управления заданиями), с помощью которого пользователь может управлять некоторыми функциями ОС, например указать, что делать с задачей, где получить необходимую информацию. Но ОС открывает пользователю новые сервисные возможности, значительно облегчающие ему работу с компьютером. Так, программа текстового редактора, заложенная в ОС персонального компьютера, позволяет пользователю редактировать любой текст при подготовке его к печати. Несложный язык редактирования осваивается без труда и почти мгновенно. Аналогичный язык имеет ОС и для построения диаграмм и графиков в программе графического редактора.

Уместно сказать, что если возможности компьютера по решению конкретных задач можно условно назвать его «образованностью», то сервисные возможности, заложенные в ОС, естественно назвать «воспитанностью». Так что современный компьютер — это не только образованный, но и воспитанный собеседник. Но чтобы получить пользу (и удовольствие) от общения с ним, надо прежде всего иметь представление о принципах организации языков общения с компьютером. Как показано выше, принципы их организации не слишком сложны, хотя овладение самими языками требует некоторых усилий.

— Послушайте, шеф, — обратился Поль к Мегрэ. — Как вы относитесь к показаниям парикмахера, заведение которого находится напротив гостиницы? Не кажется ли вам, что он слишком уж много видел? Создается впечатление, что он только и делал, что наблюдал за дверями гостиницы?

— Вы думаете, что нас водят за нос? Это в принципе возможно. И было бы очень интересно узнать, кому это нужно, ведь именно через него мы выйдем на убийцу. Но как убедиться, что он врет? Докажите мне это, а я уж вытрясу из него все, что нужно.

— Доказать едва ли смогу, а вот убедить вас попробую. И вот как. Смоделируем работу парикмахерской и посмотрим, было ли у него столько времени, чтобы разглядывать того, кто заходит в гостиницу.

— Что ж, — усмехнулся Мегрэ, — это уже конкретное предложение по применению компьютера в нашем следствии. Валяйте, но только не темните и выкладывайте все предпосылки. Я ведь хорошо понимаю, что для эффективного моделирования нужно иметь точную исходную информацию об объекте.

— Хорошо. Будем моделировать моменты прихода клиентов и время и обслуживания.

— А откуда вы узнаете, когда именно приходят эти клиенты и сколько нужно времени, чтобы их обслужить? — разочарованно заметил Мегрэ. — Не морочьте мне голову! Такой информации вам никогда и нигде не получить!

— Моменты появления клиентов в парикмахерской естестественно считать случайными. А вот свойства этой случайности можно определить, исходя из здравого смысла, например задавая ее среднее значение и интервал, в границах которого может изменяться эта случайная величина. Вы можете сказать, сколько примерно времени уходит на обработку Вашей головы?

— Ну, минут пятнадцать.

— А в каком интервале может изменяться эта случайная величина?

— Это зависит от парикмахера. Думаю, от 10 до 20 мин. А какое это имеет отношение, ведь я не обслуживался у этого парня?

— Но ведь все парикмахеры работают примерно одинаково. Поэтому среднее время обслуживания и его разброс можно считать одинаковыми для всех. Теперь о моменте появления клиентов. В налоговом управлении я узнал, что этот парикмахер принимает клиентов 25 в день. Так что можно считать, что они заходят в среднем через 18 мин, а разброс, ну скажем, плюс-минус 6 мин.

— Значит, интервал между клиентами от 12 до 24 мин. Что ж, это вполне возможно.

— Вот и все, что нужно для составления программы моделирования. Ну как, шеф, правдоподобны исходные данные для модели?

— Пожалуй, да, хотя я не исключаю возможности прихода двух клиентов одновременно, — заметил Мегрэ.

— Но сознайтесь, это редкое событие. Ведь в парикмахерскую обычно ходят поодиночке. А мы моделируем средний режим работы этого заведения — такие редкие события не играют существенной роли.

— Ну ладно. Уговорили. Как же теперь это моделировать на компьютере? — спросил Мегрэ.

— Нужно составить программу моделирования. Для этого больше подойдет язык GPSS. Используем два основных оператора этого языка: GENERATE и ADVANCE. Первый моделирует моменты появления клиентов (точнее, интервалы между их появлением), а второй — время обслуживания клиентов. Числа, стоящие вслед за операторами, определяют свойства потока клиентов и времени обслуживания. Так,

GENERATE 18, 6, 10, 25

означает, что клиенты появляются в среднем через 18 мин, причем сам момент появления случаен и находится в интервале (18 ± 6) мин, т. е. от 12 до 24 мин,

которые мы согласовали. Первый клиент приходит через 10 мин после открытия парикмахерской в интервале (10±6) мин, т. е. от 4 до 16 мин. Последняя цифра определяет число клиентов, которые нужно моделировать (это нагрузка парикмахерской за один день). Приоритетов учитывать не будем.

— Что, этим будет моделироваться работа одного дня? — спросил Мегрэ. — Но ведь один день ничего не определяет.

— Но определяет загрузку парикмахера в этот день, — перебил его Поль. — А потом сможем и повторить моделирование для любого числа дней. Но начнем с одного дня работы.

Теперь об обслуживании. Оно моделируется оператором задержки так:

ADVANCE 16, 4

т. е. время обслуживания случайное (16±4) мин — от 12 до 20 мин, а в среднем 16 мин.

— И это все. Неужели так просто? — изумился Мегрэ.

— Не торопитесь. Если бы язык был предназначен только для моделирования таких простых задач, то этим можно было бы ограничиться. Но язык GPSS универсальный и поэтому обладает избыточностью, необходимой для моделирования других систем. Поэтому программа моделирования кроме указанных операторов будет содержать и другие, устраняющие избыточность языка. Вот эта программа. И Поль быстро написал следующее и прокомментировал:

SIMULATE	Нужно моделировать (симулировать) программу, а не просто просмотреть ее, например для отыскания компьютером ошибок программирования
GENERATE 18, 6, 10, 25	Появление клиентов (этот оператор мы уже рассматривали выше)
QUEUE 1	Стать в очередь № 1. С помощью этого оператора пришедший клиент присоединяется к очереди № 1. Здесь других нет, но большое число очередей предусмотрено в языке GPSS. Вот и приходится нумеровать одну очередь, хотя других нет и в помине. Это и есть устранение избыточности языка.
SEIZE A	Занять место A (свободное кресло первым клиентом из очереди)
DEPART 1	Покинуть очередь № 1. Этот оператор логично было бы поставить перед предыдущим (сначала покинуть очередь, а затем занять кресло). Но так уж устроен язык GPSS. Так ему проще.
ADVANCE 16, 4	Задержка на обслуживание (мы уже знаем этот оператор)
RELEASE A	Освободить место A (освобождение парикмахера)
TERMINATE	Завершить (Уход клиента из парикмахерской. Это не формализм, а важный момент ухода клиента из системы. В другой задаче этот

клиент мог бы отправиться на следующее место обслуживания, например к маникюрше. Здесь этого не нужно)

— Вот и все! Теперь осталось ввести эту программу в память компьютера, вызвать транслятор с языка GPSS и перевести с его помощью программу на машинный язык. Полученная программа и будет выполняться компьютером.

Поль быстро набрал на клавиатуре программу и запустил ее. Через несколько секунд на экране дисплея появилось:

472 минуты	Это означает, что с открытия парикмахерской до момента ухода 25-го клиента прошло 7 ч 52 мин
Загрузка А: 86%	Парикмахер был загружен 86 % своего времени, а остальные 14 % ожидал клиента и мог разглядывать вход в гостиницу. На это у него было примерно 66 мин
Максимальная очередь: 1	Очередь не превышала одного человека
Среднее время ожидания обслуживания: 5, 13 мин и т. д.	В среднем клиент ожидал обслуживания 5 мин 8 с

— Все эти данные вычисляются программой автоматически, и их не нужно программировать, — заметил Поль.

— Из всего этого самое интересное — 66 мин свободного времени парикмахера. Это чуть больше часа. А для наблюдения всех приведенных им сведений нужно не менее 3 часов. Но ведь день на день не приходится и, может быть, именно в тот день у него были эти 3 часа. Можно ли определить, как часто ему удается отдыхать три часа в день?

— Можно. Будем «крутить» программу одного дня до тех пор, пока свободное время не составит 30 % от рабочего дня или больше.

Поль добавил несколько операторов к программе. Через несколько минут компьютер остановился и выдал на экран дисплея:

ПРОГОН № 102

— Это что же, — изумился Мегрэ, — из 100 дней лишь один день он мог 3 часа наблюдать за гостиницей?! Ему можно верить с вероятностью лишь в одну сотую! Это значит, что его показания лживы.

— Выходит так, — согласился Поль.

— Немедленно вызовите мне его и выведите на печать результаты моделирования. Не думаю, что ему удастся долго запираться перед такими аргументами.

— Слушаю, шеф. Только не слишком надейтесь, что он поверит компьютеру.

— Не поверит компьютеру — поверит мне. Ведь допрашивать его будет не компьютер, — усмехнулся Мегрэ.

6. ВСЕМ МИРОМ
(Вычислительные системы)

СПЕЦИАЛИЗАЦИЯ ПРОТИВ УНИВЕРСАЛЬНОСТИ

До сих пор мы рассматривали универсальные ЭВМ, создаваемые для решения очень широкого круга задач обработки информации. Идея универсальности ЭВМ с изменяемой программой была настолько нова и необычна, что на первых порах развития вычислительной техники проектировщики и думать об ином не хотели. Да и компьютеров было слишком мало, чтобы специализировать их на какие-то конкретные задачи.

По мере применения компьютеров для решения разнообразных задач, число которых росло чрезвычайно быстро, стало ясно: универсальность это хорошо, но и некоторая специализация неплохо, а иногда и лучше универсальности. Дело в том, что задачи, решаемые компьютером, довольно естественно разбиваются на вычислительные задачи и задачи моделирования. Специфика их и породила различные языки программирования.

Всякая универсальность хороша только своей универсальностью и не более. А с решением конкретных задач специализированная ЭВМ справится всегда лучше, ведь она создана для решения этих задач. Очевидно, что и при обработке информации есть важные задачи, которые требуют для своего решения специальных компьютеров. Это не значит, что эти задачи не могут решаться на универсальной ЭВМ, но сделает она это хуже, чем специализированная. Именно это обстоятельство и заставляет создавать специализированные компьютеры, которые обеспечивают эффективное решение задач определенного класса.

Но всякая специализация приводит к расширению одной возможности за счет сужения других. Вспомним известный афоризм Козьмы Пруткова: «Всякий специалист подобен флюсу — его полнота односторонняя». Именно такая односторонность и характерна для специализированных ЭВМ. Какие же возможности расширяют они? Их много. Например, необходимость распределения больших вычислительных мощностей между многими пользователями привела к созданию вычислительных *систем коллективного пользования*, которые предоставляют каждому пользователю возможность выходить со своего пульта на высокопроизводительную машину. Эта система отличается от обычной ЭВМ высокой производительностью, большим числом устройств ввода-вывода и специальными средствами разделения машинного времени между пользователями, чтобы они не мешали друг другу. Но особенно много вычислительных систем создано для решения информационно-поисковых задач. Их так и называют — информационно-поисковые системы (ИПС). Они отличаются большой памятью и оперативными средствами доступа к содержимому. Для управления реальными

объектами (например, технологическими процессами) также необходимы специализированные компьютерные системы. К ним предъявляются свои требования: высокая производительность и большая надежность.

Всякая четко сформулированная цель специализирует компьютер. Грубо говоря, сколько целей, столько и должно быть специализированных компьютеров. Но это, пожалуй, слишком расточительно. Поэтому стоит говорить лишь об основных целях, которые определяют облик компьютера. А их немного ... Всего две.

Все задачи, решаемые человечеством, условно и довольно грубо могут быть разделены на два больших класса: задачи, которые надо решить очень быстро, и задачи, которые надо решить очень надежно, без ошибок. Это, конечно, не значит, что при быстром решении не нужна надежность, а на надежное решение можно тратить очень много времени. Просто в разных задачах эти две противоречивые цели играют разную роль: в одних основным является время решения, а в других — надежность.

Задачи первого класса требуют большой производительности вычислительных систем, а второго — высокой надежности. Что же это за задачи?

ЗАЧЕМ НУЖНА ВЫСОКАЯ ПРОИЗВОДИТЕЛЬНОСТЬ?

Высокая скорость вычислений нужна для удовлетворения двух насущных потребностей человечества — решения сложных вычислительных задач и обработки больших потоков информации.

К первому классу сложных вычислительных задач относятся такие важные задачи, как составление народнохозяйственных планов, решение научных задач, возникающих в аэродинамике (например, при исследовании процесса обтекания газом профиля самолета или ракеты), в сейсмологии (например, при определении строения Земли по результатам сейсмических наблюдений), в метеорологии (например, при составлении прогнозов поведения атмосферы), в физике плазмы и т. д. Все эти задачи требуют для своего решения огромного количества вычислительных операций — порядка 10^{10} — 10^{12} на задачу. Обычный компьютер (производительностью примерно 10^6 операций в секунду) решал бы эти задачи годами и десятилетиями. Но ответ нужен сейчас, сегодня, через час! Это и заставляет создавать *суперкомпьютеры* — так называют ЭВМ производительностью не менее чем 100 млн. операций в секунду. Каждый такой суперкомпьютер имеет вычислительную мощность, более чем в 100 раз превышающую мощность обычного компьютера. Очевидно, что и устроен он должен быть иначе.

Другой источник задач, требующих больших вычислительных

мощностей, порожден потребностью обрабатывать мощные потоки данных.

Эти потоки данных порождаются различными источниками информации. Но, пожалуй, самым мощным источником являются многочисленные датчики информации, расположенные на изучаемом или управляемом объекте, например в космосе, атмосфере, на земле и под ней, при научном эксперименте, на производстве, в больнице и т.д. Успех обычно зависит от быстроты обработки этих данных, которые следует перерабатывать со скоростью их поступления, в том темпе, в котором они поступают.

Современные средства сбора информации обладают огромной производительностью. Именно поэтому необходимо создавать вычислительные средства, способные перерабатывать потоки данных в режиме реального времени — в темпе их поступления, иначе необработанные данные будут неограниченно накапливаться и обесцениваться ввиду их неизбежного старения (особенно для задач оперативного управления). Вычислительные системы для обработки потоков данных называют *системами реального времени*, подчеркивая их зависимость от окружающей реальности, от внешней среды.

Если работа обычного компьютера определяется только его программой, то работа вычислительных систем реального времени зависит еще и от потока данных, и прежде всего от интенсивности этого потока, который нужно обрабатывать со скоростью его поступления в реальном времени. Именно для этого нужна высокая производительность суперкомпьютера — ему приходится «перемалывать» огромные потоки информации, поступающие от многочисленных источников. Смело можно сказать, что на любой суперкомпьютер найдется такой поток данных, с которым он не справится. Поэтому так важно создание вычислительных систем высокой и сверхвысокой производительности для обработки данных в реальном масштабе времени.

КАК ПОВЫСИТЬ БЫСТРОДЕЙСТВИЕ?

Очевидно, что производительность компьютера можно повысить, увеличив скорость работы его процессора. И в этом направлении ведутся интенсивные работы. Но каждый шаг в этом направлении дается с большим трудом. Действительно, быстродействие процессора ограничено скоростью распространения электрического сигнала, которая не может быть больше скорости света (300 тыс. км/с). Следовательно, для повышения его производительности надо уменьшить размеры процессора. Но и их нельзя делать произвольно малыми. Современная микроэлектронная технология позволяет сделать процессор на кристалле в несколько квадратных миллиметров, но ... не менее. Теоретические расчеты показывают, что с учетом дальнейшего развития

микроэлектронной технологии производительность процессора не может быть более 300 млн. операций в секунду. Это теоретический предел! А сейчас коммерческие процессоры достигают производительности в 1 — 10 млн. операций в секунду. И именно на них следует ориентироваться при создании суперкомпьютера.

Как же, располагая сравнительно «медленными» процессорами, достичь сверхвысокой производительности вычислительных систем? Ответ здесь один...

Распараллеливание

Это одновременно решение разных частей одной и той же задачи. Пусть нужно решить задачу, требующую $N = 10^{10}$ машинных операций, за $T = 100$ с. Очевидно, что для этого необходимо взять одну ЭВМ производительностью N/T операций в секунду. Но таких машин не бывает. Как же быть? Только распараллеливать! Пусть для этого мы располагаем процессорами производительностью $V = 10^6$ операций в секунду. Тогда для решения этой задачи в указанное время нужно как минимум $N/(TV) = 100$ таких процессоров, если, разумеется, задача допускает распараллеливание, т. е. ее можно разбить на независимые части, которые будут решаться одновременно на разных процессорах. Если это удастся сделать и загрузить решением одновременно все 100 процессоров, т.е. получить производительность 10^8 операций в секунду, то наша задача будет решена за требуемые 100 с. Но для этого, как видно, нужно суметь распараллелить процесс решения задачи так, чтобы все 100 процессоров работали без простоев. Это редко когда удается и лишь для задач, составленных из независимых подзадач (для нашего примера таких подзадач должно быть 100 и все они должны быть одинаковыми, т.е. иметь трудоемкость по 10^8 операций).

Очевидно, что в других случаях понадобится еще больше процессоров. Причем вся трудность состоит в том, чтобы запрограммировать сам вычислительный процесс — указать, когда и какую часть задачи должен решать каждый процессор, откуда он должен получить исходную информацию (данные) и куда отправить полученный результат. От качества такой программы зависит эффективность решения задачи, т. е. требуемый ресурс — число процессоров, необходимое для решения поставленной задачи в заданное время.

Но не следует думать, что составить такую программу очень сложно. Дело в том, что время решения задачи зависит от загрузки этой задачей всех имеющихся в системе процессоров. И, как не странно, рецепт создания программы управления вычислительным процессором довольно прост. Надо стараться максимально загрузить каждый процессор, а значит, минимизировать его простои и работу на решение вспомогательных задач (напри-

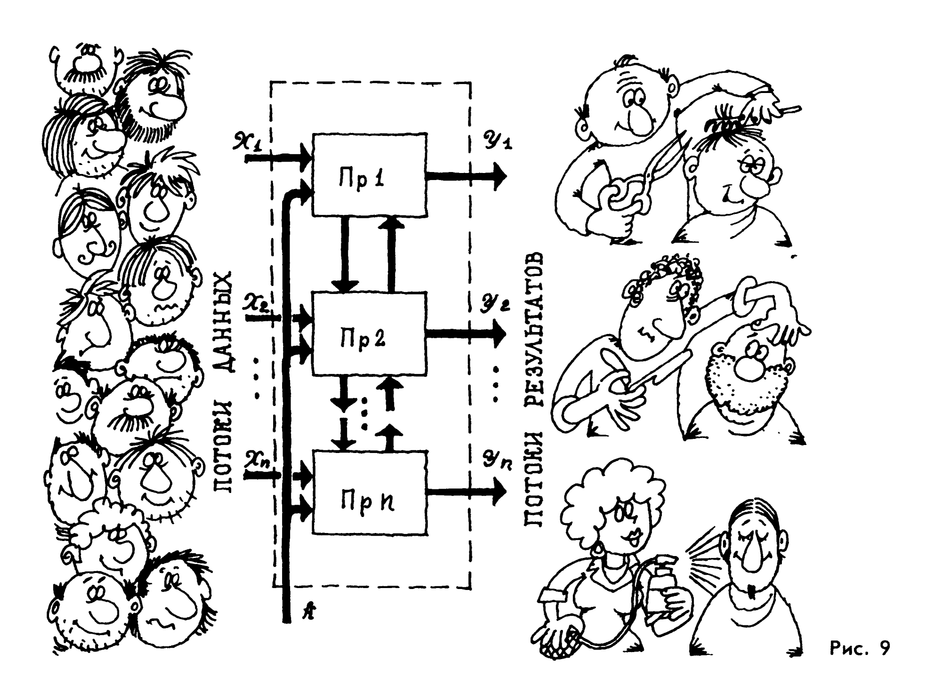


Рис. 9

мер, на обмены информацией между процессорами, которые, вообще говоря, неизбежны). Так обычно и поступают. Поэтому все искусство программирования при решении задач на *мультипроцессорной* вычислительной *системе* (именно так называют систему со многими процессорами) заключается в обеспечении минимальных временны́х затрат на простои процессоров и обмены данными между ними.

Структура такого мультипроцессора может быть самой разнообразной, более того, она может изменяться в процессе решения. Но прежде всего она зависит от решаемой задачи. Если задача легко членится на слабозависимые части, то следует применять так называемую параллельную обработку, которая обеспечивается набором из параллельно работающих процессоров (рис. 9). Система из n взаимосвязанных процессоров (Пр1... Пр n) обрабатывает n потоков данных $x_1,..., x_n$, преобразуемых в n потоков результатов $y_1,..., y_n$. Связи между процессорами позволяют им обмениваться необходимой промежуточной информацией. Такую систему параллельной обработки обозначают часто ОКМД — Одиночный поток Команд (k) и Множественный поток Данных ($x_1,..., x_n$). Одиночность потока команд заключается в том, что все процессоры выполняют одновременно только одну команду, затем другую и т.д. (на рис. 9 каналы передачи команд изображены косыми стрелками).

Эту схему обработки часто называют векторной, а такой

мультипроцессор — векторным процессором, так как с его помощью очень удобно обрабатывать n-мерные векторы, например траекторию движения летательного аппарата (в этом случае $n = 3$, если описывается движение его центра тяжести, и $n = 6$, если описывается движение твердого тела).

Такой способ распараллеливания естественно назвать *распараллеливанием* в *пространстве,* так как разные части одной и той же задачи решаются на разных процессорах, т. е. распределены в пространстве. Лучше всего этим способом решаются задачи со слабо связанными частями, что требует небольшого числа обменов информацией между процессорами и поэтому незначительно снижает производительность обработки, приближая ее к теоретической. А теоретическая производительность равна сумме производительностей всех процессоров системы. Она реализуется лишь для задач, все n частей которой не связаны друг с другом и имеют одинаковую трудоемкость.

Но очевидно, что далеко не все задачи имеют такую слабосвязанную структуру. Моделью программы задачи с сильносвязанной структурой является такая последовательность операторов, при которой каждый оператор исходные данные получает от предыдущего, а результат обработки направляет следующему оператору. Здесь ничего в пространстве распараллелить нельзя. Как же быть?

Если такую задачу нужно решать только один раз, то ничего не поделаешь и распараллелить ее решение не удастся. Но если ее надо решать многократно при различных исходных данных, то распараллеливать можно, но ... во времени.

Идея *распараллеливания во времени* известна давно и широко используется в массовом производстве. Это идея конвейе-

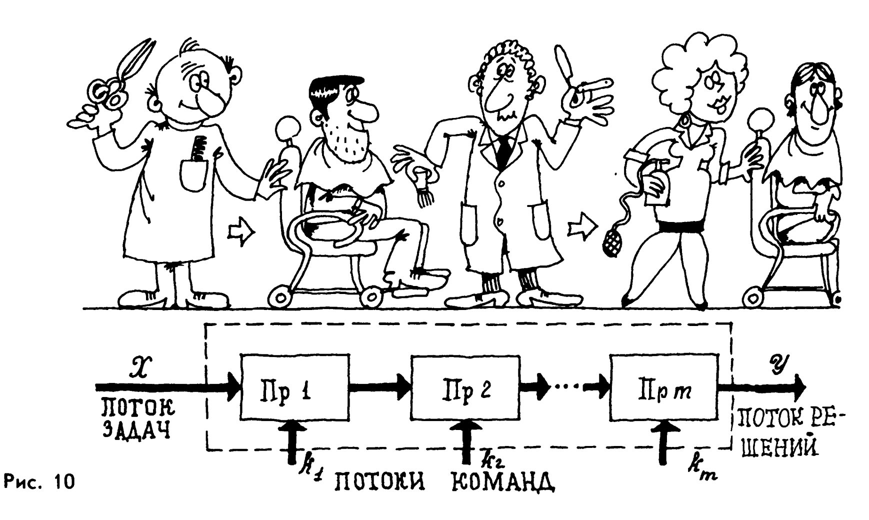


Рис. 10

ра (рис. 10). На вход x подается поток исходных данных (задач) и каждый i-й процессор работает по своему потоку команд k_i. Задача, таким образом, решается сначала на первом процессоре, затем на втором и т. д. А в это время на первом решается другая задача при других начальных условиях и т. д. В результате одновременно в конвейере решается столько задач (с различными начальными условиями), сколько процессоров в этом конвейере. При этом каждая задача в процессе решения продвигается по конвейеру. Распараллеливание здесь происходит не только в пространстве (т. е. по разным процессорам), но и во времени.

Как видно, конвейер процессоров реализует Множественный поток Команд и Одиночный поток Данных — МКОД. Такая схема получила название *конвейерной обработки*. Отличается она тем, что при обработке одного потока данных одновременно выполняются сразу несколько команд из разных потоков команд. Эта схема обработки очень удобна при работе с программой, которую нельзя разбить на независимые части, но эти части связаны лишь через данные, которые обрабатывает эта программа.

Легко заметить глубокую связь этого вида обработки с промышленным конвейером, где роль процессоров играют рабочие места, а данными являются заготовки. Так же, как у промышленного конвейера, производительность конвейерной обработки определяется числом и трудоемкостью операций, выполняемых каждым процессором: чем они меньше, тем производительнее работает конвейер.

Матричная обработка

Так как векторная и конвейерная обработка обеспечивают высокую производительность лишь для специфических задач, то естественно их объединить для решения задач разного рода. Это делает матричная схема обработки — гибрид векторной и конвейерной схем. При этом реализуется самая сложная формула МКМД — Множественный поток Команд и Множественный поток Данных (рис. 11). Такой мультипроцессор (он обведен штриховой линией) называется матричным: по своей структуре он напоминает матрицу (таблицу).

Примером матричной вычислительной системы является ИЛЛИАК-IV (США). Она состоит из 64 процессорных элементов (ПЭ), что обеспечивает общую производительность до 200 млн. операций в секунду. У нас матричную структуру может иметь перестраиваемая вычислительная система ПС-3000. Она содержит 64 ПЭ и имеет производительность на специальных задачах до 100 млн. операций в секунду.

Следует отметить, что последняя из возможных формул ОКОД — Одиночные потоки Команд и Данных — соответствует

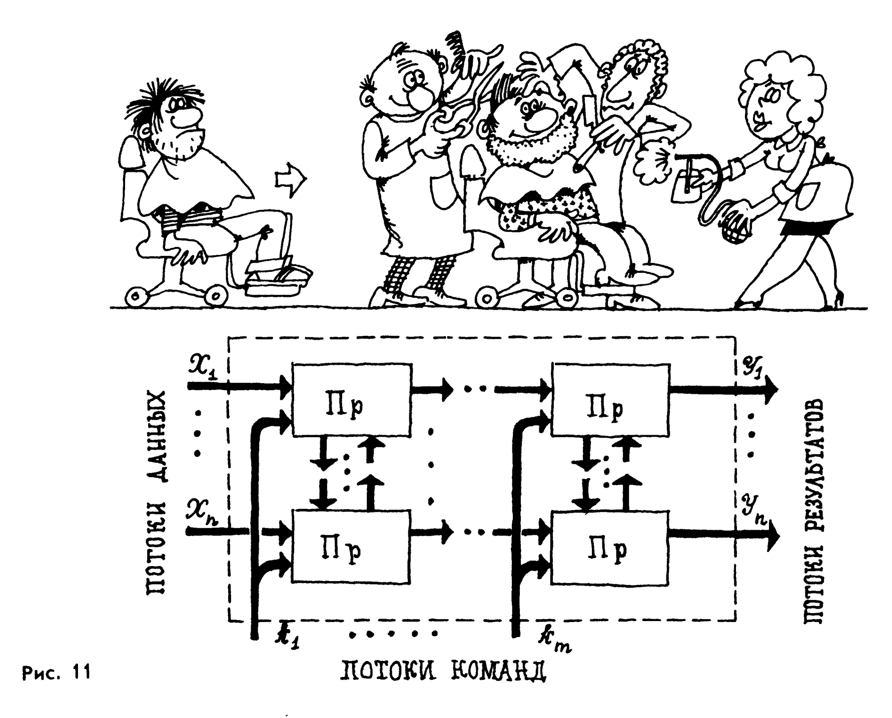


Рис. 11

обычной однопроцессорной ЭВМ, обрабатывающей один поток данных, которая рассмотрена в гл. 1.

АРХИТЕКТУРА КОМПЬЮТЕРА

Говоря о различных вычислительных системах, нельзя не упомянуть о таком очень распространенном понятии, как архитектура вычислительной системы. Это слово заимствовано из зодчества. Там оно означает потребительские характеристики строительного объекта, т. е. те, которые важны жильцам. Например, число и размер комнат, их взаимное расположение, высота потолков определяют архитектуру квартиры. Именно эти характеристики являются важнейшими для жильцов, а не способ введения коммуникаций в квартиру, что волнует эксплуатационников жэка, но не жильцов. Именно архитектура квартиры дает возможность оценить эффективность ее использования конкретной семьей со своим составом, возрастом ее членов, их потребностями и причудами. Совершенно аналогично под архитектурой вычислительной системы понимают совокупность ее свойств и характеристик, которые должен знать пользователь для эффективного использования вычислительной системы при решении своих задач. Что же это за свойства и характеристики?

Прежде всего, это правила представления и передачи информации при взаимодействии пользователя и системы, т. е. коды представления данных и машинные языки. Другим важным свойством архитектуры являются сопряжения основных устройств вычислительной системы — ее структура. Но архитектуру не следует путать со структурой. Архитектура определяет правила взаимодействия составных частей системы, регламентирует не все связи структуры, а лишь наиболее важные, т. е. те, которые нужны для грамотного использования компьютерной системы. Очевидно, что векторный, конвейерный и матричный тип мультипроцессоров образуют соответствующие типы архитектур вычислительных систем. Они имеют свои достоинства и недостатки, которые следует учитывать при решении тех или иных задач так, чтобы эффективно использовать достоинства и надежно нейтрализовать недостатки различных архитектур.

МАШИНЫ ПОТОКА ДАННЫХ

Выше мы рассматривали различные способы параллельной обработки информации, позволяющие значительно повысить производительность компьютера при обработке данных. Однако совершенно не обязательно, чтобы данные поступали извне в компьютер. Они могут генерироваться в самой машине как промежуточные результаты. Это позволяет использовать описанные принципы параллельной обработки для решения громоздких вычислительных задач и при отсутствии потока данных извне.

Схема обработки представлена на рис. 12, где мультипроцессор (МП) реализован по матричной схеме (рис. 11). Здесь потоки промежуточных результатов (данных) $x_1,...,x_n$, получаемые в процессе решения задачи, являются исходными данными для последующих вычислений. Потоки команд $k_1,...,k_m$ обеспечивают обработку поступающих на МП потоков данных, а потоки результатов y_1, ..., y_n поступают в ОЗУ, чтобы стать потом потоком данных.

Такая схема решения задачи обладает очень большой гибкостью. Здесь реализуется процесс вычислений, который управляется не программой, а потоком данных, получаемых в процессе вычислений. Это новый принцип управления, когда данные управляют процессом обработки, осуществляется он довольно просто: нужно выделять свободный процессор той части программы, для которой имеются данные.

На первый взгляд это очевидно. Не выделять же процессору задачу, не обеспеченную данными! Но последовательное применение этого подхода приводит к тому, что заранее уже нельзя сказать, какой обработкой будет занят тот или иной процессор,— все решится после его освобождения от предыдущей работы. И ему в соответствии с принципом управления данными поступит зада-

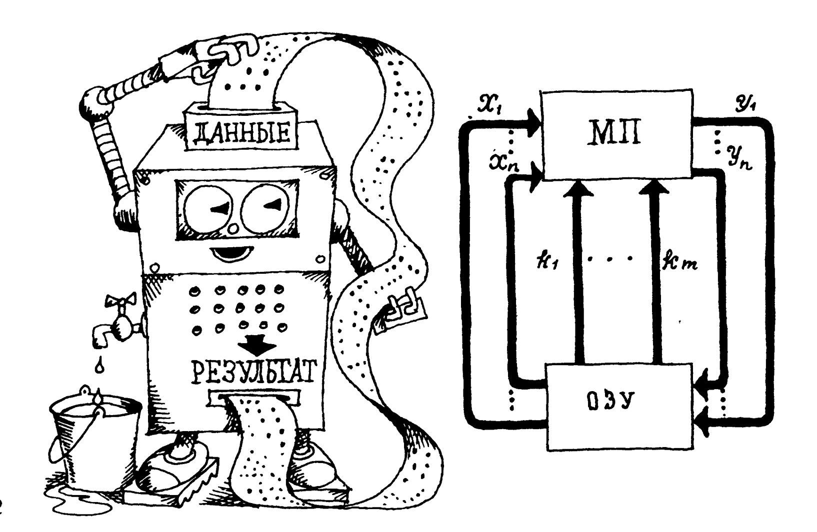


Рис. 12

ча, для которой к этому моменту будут готовые данные. Именно так работает машина потока данных.

Так данные (точнее, моменты их появления в вычислительной системе) управляют вычислительным процессом. Легко заметить, что чем больше процессоров, тем более эффективно применение этого подхода, так как меньше вероятность, что процессоры будут простаивать в ожидании данных и данные в ожидании свободного процессора. Именно поэтому эффективность машины потока данных тем выше, чем больше процессоров она имеет.

КОМПЬЮТЕР ИЗ... КОМПЬЮТЕРОВ

Описанные архитектуры многопроцессорных вычислительных систем могут реализоваться в помощью нескольких однопроцессорных ЭВМ. В этом случае получаются *многомашинные* вычислительные *системы*. Но при этом следует помнить, что для соединения двух и более ЭВМ требуются специальные согласующие средства — адаптеры, что усложняет задачу.

Но, пожалуй, самыми распространенными средствами параллельной обработки являются *вычислительные комплексы*. Они образуются из имеющихся компонентов вычислительных машин путем комплексирования — объединения их в единый комплекс. В качестве таких компонентов выступают процессоры, оперативная память (ОЗУ), внешняя память (ВЗУ), каналы связи между компонентами и т. д. Всего таких компонентов с учетом различных типов и модификаций может быть сотни и тысячи. Так что процесс комплексирования системы по заданным требованиям достаточно сложен. Но зато и велик эффект. Сейчас для решения конкретных задач редко кто приобретает ЭВМ в стандартном испол-

нении — обычно берут ее компоненты со средствами комплексирования: адаптерами, пультами для изменения конфигурации системы, устройствами управления комплексом и т. д. Примерами таких комплексов являются выпускаемые нашей промышленностью ВК-1010 (на базе двух ЭВМ ЕС-1030), ВК-2Р-50 (на базе двух ВМ ЕС-1050), ВК-2М45 (на базе двух ЭВМ ЕС-1045) и т. д.

АССОЦИАТИВНАЯ ОБРАБОТКА

Однако производительность обработки информации вычислительной системы определяется не только работой ее процессоров. Значительное время затрачивается на поиск информации в памяти системы, что, естественно, снижает ее производительность. Дело в том, что поиск информации связан с очень распространенной задачей принятия решения в сложившейся ситуации. Заранее известно (задано, вычислено), что в определенной ситуации S надо действовать заданным образом R — это и есть решение. Ситуация и решение кодируются и запоминаются в одной ячейке памяти. Если таких ситуаций (и, соответственно, решений) немного, то особой проблемы нет — можно просмотреть подряд все ячейки памяти, определить, какая их них содержит ситуацию S, совпадающую или близкую с той S′, которая сложилась, и принять решение R, соответствующее ситуации S′. Но если таких ситуаций много и принимать решение нужно часто, то последовательный просмотр всех ячеек памяти займет слишком много времени, что резко снизит производительность всей системы. Как же быть?

Эту задачу решает так называемое ассоциативное запоминающее устройство (ЗУ), позволяющее обращаться к памяти не по номеру ячейки, как в обычном ЗУ, а по ее содержимому. Напомним, что обычно оперативное запоминающее устройство (ОЗУ) всякого компьютера устроено так, что оно выдает содержимое ячейки в ответ на запрос, сформулированный в виде номера (адреса) этой ячейки. Такое ОЗУ называют адресным.

При ассоциативной обработке на вход ассоциативного ЗУ подается двоичный код, описывающий сложившуюся ситуацию S′, в которой необходимо принять решение. Этот код поступает на все ячейки ассоциативного ЗУ одновременно и сравнивается с их содержимым. Но так как в ячейках записаны не только ситуации, но и решения, то следует исключить из рассмотрения (замаскировать) ту часть ячейки, где нет описания ситуации.

Делается это с помощью *механизма маски*, который позволяет выделять и сравнивать только ситуации. Действует он следующим образом. Вместе с ситуацией в ассоциативное ЗУ поступает и маска — двоичный код, имеющий длину, равную числу ячеек в памяти. Тот разряд, в котором маска имеет нуль, исключается из сравнения. Маска своими нулями как бы закрывает (маскирует) все,

что не относится к ситуации. В результате сравниваются только те разряды кодов, которые разрешены маской, т. е. имеют в ней единицы.

Пусть, например, в трех 12-разрядных ячейках ассоциативного ЗУ записана информация о ситуации S (первые 8 разрядов) и решения R (последние 4 разряда):

$$
\begin{array}{rl}
\text{ЗУ} \left\{ \begin{array}{l} A1 = 1001\ \ 1101\ \ 1101 \\ A2 = 1001\ \ 1010\ \ 1010 \\ A3 = 1001\ \ 1001\ \ 0111 \end{array} \right. & \\
\text{Маска:}\ \ M = 1111\ \ 1100\ \ 0000 & \\
\text{Ситуация:}\ S' = 1001\ \ 1000 &
\end{array}
$$

В этом случае две ячейки (A2 и A3) дают совпадение с замаскированной ситуацией S′.

Из полученных двух решений нужно выбрать одно. Это делается путем уменьшения маски, т. е. увеличения числа ее единиц. При M = 1111 1110 0000, как легко видеть, получаем одну совпадающую с ситуацией S′ ячейку A3 (при учете маски, разумеется), откуда и извлекается решение (оно было замаскировано) R3 = = 0111, которое и принимается в сложившейся ситуации S′.

Таким образом, процесс функционирования ассоциативного ЗУ сводится к следующему. На вход ассоциативного ЗУ подаются код ситуации и маска, указывающая, какие именно признаки (разряды) этой ситуации являются существенными. По этим признакам и производится одновременное сопоставление ситуации с содержимым всех ячеек. При совпадении замаскированной ситуации с хранимой в ячейке эта ячейка посылает сигнал в управляющее устройство, которое и выбирает все содержимое этой ячейки, где содержится искомая информация о решении (она была замаскирована в процессе поиска). Если таких сигнализирующих ячеек окажется много, то управляющее устройство может увеличить число существенных признаков, пропускаемых маской, и тем самым уменьшить число таких ячеек.

Преимущество такого способа в параллельности, одновременности обращения ко всем ячейкам памяти. Чем больше ячеек памяти, тем больше эффективность ассоциативного ЗУ по сравнению с адресным, Так, оно может работать в 100, 1000 и более раз быстрее адресного ЗУ, если имеет 100 и 1000 ассоциативных ячеек памяти.

Наличие управления маской в ассоциативном ЗУ делает его гибкой системой обработки информации. Именно поэтому такие ЗУ называют *ассоциативными процессорами*. Эти процессоры нашли широкое применение для управления сложными объектами.

Примером такого применения ассоциативных систем обработки является контроль за положением самолетов в зоне аэропорта, где ячейки ассоциативного ЗУ заполняются координатами всех самолетов. При этом ассоциативный процес-

сор для каждого самолета определяет ближайший к нему и, если расстояние между ними станет опасным, сообщает об этом диспетчеру. Делается это следующим образом. Ячейки ассоциативного ЗУ имеют три сегмента, где записываются координаты самолета, находящегося в зоне аэропорта. Эти координаты сообщает локатор в виде двух углов, определяющих направление на самолет и расстояние до него. Информация эта обновляется с каждым оборотом антенны локатора. Ассоциативный процессор выбирает по порядку координаты одного из самолетов и подает на вход ассоциативного ЗУ с соответствующей маской. Эта маска подобрана так, что выделяет лишь те самолеты, которые находятся в опасной близости. Решением в данном случае является совпадение всех незамаскированных координат. Оно и сообщается диспетчеру. Как видно, такой ассоциативный процессор «молчит» при нормальной обстановке в зоне аэропорта и сигнализирует лишь при возникновении опасного сближения самолетов. Причем от момента возникновения аварийной ситуации до ее выявления компьютером проходит минимальное время, так как проверка близости одного из самолетов ко всем остальным, находящимся в зоне аэропорта, происходит за один такт обращения к ассоциативному ЗУ компьютера. Меньше уже нельзя!

Ассоциативная обработка позволяет значительно повысить производительность компьютера. Это повышение грубо можно оценить числом ячеек ассоциативного процессора. Именно поэтому ассоциативная обработка нашла свое применение в системах оперативного управления (таких, как описанная выше система управления аэропортом) и в компьютерах сверхвысокой производительности, за которыми закрепилось название ...

СУПЕРКОМПЬЮТЕРЫ

Все описанные выше приемы распараллеливания обработки информации направлены на повышение производительности компьютера. Эта производительность обычно измеряется так называемым *флопами* — количеством операций сложения с плавающей точкой в одну секунду (это самая распространенная операция). Компьютер, имеющий производительность свыше 100 Мфлоп (мегафлоп), т. е. 10^8 таких операций в секунду, называют суперкомпьютером.

Одним из типичных представителей и лидером этого направления является суперкомпьютер Cray-2, имеющий производительность 1000 Мфлоп и оперативную память емкостью свыше $2 \cdot 10^8$ 15-разрядных слов — огромная производительность, требующая гигантской оперативной памяти. У нас направление суперкомпьютеров возглавляет ЭВМ «Эльбрус-2», имеющая 10 высокопроизводительных процессоров, каждый из которых может работать со скоростью свыше 10 Мфлоп.

Сейчас в мире сравнительно немного суперкомпьютеров, всего несколько сотен — капля в море по сравнению с сотнями миллионов персональных ЭВМ. Они крайне дороги, громоздки и нуждают-

ся в специальных программах. Напомним, что суперкомпьютер — это многопроцессорная вычислительная система, производительность которой получена за счет глубочайшего распараллеливания процесса вычислений по всем процессорам системы. Реализовать такое распараллеливание помогает искусство программиста, от которого зависит производительность суперкомпьютера при решении той или иной задачи. Создание программы для суперкомпьютера требует не только искусства, но и больших временных затрат. Именно поэтому при решении специальных задач иногда приходится отказываться от услуг универсального суперкомпьютера, а создавать специальные компьютеры, моделирующие интересующие процессы. Например, для расчета орбит взаимодействующих планет Солнечной системы был создан специализированный компьютер Digital Orrery. Работая со скоростью 10 млн. операций в секунду, он позволил определить положение пяти внешних планет Солнца за 100 млн. лет до и после нас.

Но вернемся к универсальным суперкомпьютерам и рассмотрим некоторые задачи, возлагаемые на них. Одной из таких задач является создание изображений. Это направление называют ...

Машинная графика

Каждый конструктор знает, как важно в процессе проектирования иметь хорошее представление о форме проектируемого изделия. Три вида детали, изображенные на стандартном чертеже (главный, слева и сверху), дают полную информацию о детали, но ... не обладают необходимой наглядностью для человека (рис. 13, *а*). Для преодоления этого недостатка давно придумано так называемое изометрическое изображение, показывающее деталь как бы чуть сбоку и сверху (рис. 13, *б*). Достаточно одного взгляда на такое изображение, чтобы получить представление о форме детали. Еще лучше, если это изометрическое изображение можно поворачивать, как бы обходить вокруг детали или вращать ее перед собой (рис. 13, *б-д*).

Алгоритм построения изометрического изображения по чертежу детали несложен (этому учат на первом курсе в каждом техническом вузе), но очень громоздок. И чем сложнее деталь, тем более громоздкими становятся расчеты изометрического изображения, которые нужно повторять в полном объеме для построения каждого нового ракурса.

Именно эти расчеты и возлагаются на компьютер, который полученный результат выводит на экран дисплея в виде изометрического изображения. Чтобы создать эффект вращения детали, достаточно 25 раз в секунду изменять ракурс на малый угол и строить новое изображение. Несложный расчет показывает, что для такой операции понадобится огромная производительность компьютера, который должен «вычислить» каждую точку изомет-

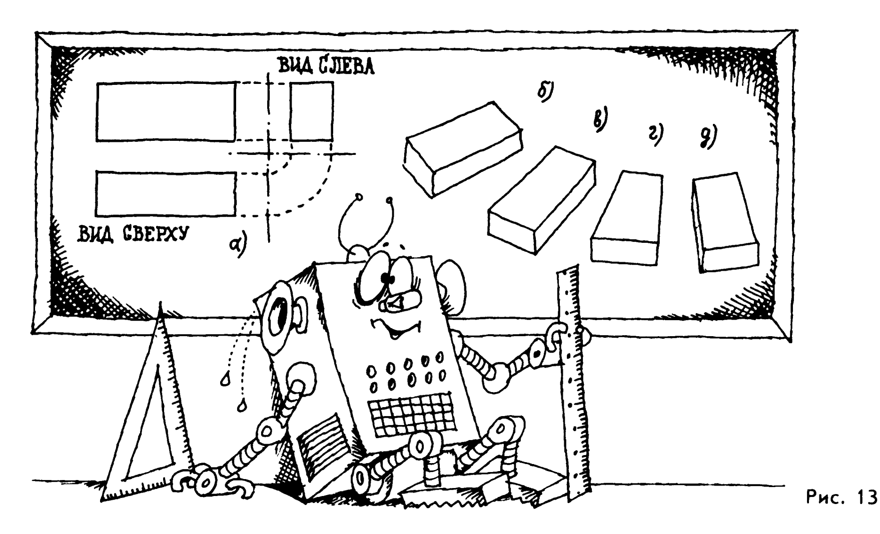


Рис. 13

рического изображения 25 раз в секунду. А этих точек на экране около миллиона! Вот и получается, что с этой задачей справляется лишь суперкомпьютер! (Заметим, что, строго говоря, не приходится обсчитывать все точки изображения: поверхность детали формируется из плоских участков, внутренность которых окрашивается равномерно. Так, для изображения кирпича, показанного на рис. 13, *б-д,* понадобится лишь три таких участка. Если же деталь имеет криволинейные поверхности, то размер плоских участков, образующих эту поверхность, будет очень малым, а их число соответственно большим. Так что для сложных форм вычислительные затраты на построение изображения всегда очень велики и требуют суперкомпьютера для своей реализации.)

Свое практическое применение такого рода суперкомпьютерные системы машинной графики нашли в системах автоматизированного проектирования (САПР). С их помощью проектируются конструкции самолетов, ракет, автомобилей и других сложных изделий современной техники.

Можно ли пощупать воображаемую деталь?

Оказывается можно! И обеспечивает это все тот же суперкомпьютер. Здесь у дотошного читателя возможно возникнет коварный вопрос: «А зачем щупать то, чего не существует?» Во-первых, это просто интересно. Действительно, возможность ощупывать воображаемый и представленный компьютером на экране дисплея мир открывает совершенно новую страницу в познании.

В одном из научно-фантастических рассказов 70-х годов

описывались так называемые онирофильмы, в которых сочетались зрительные и тактильные ощущения. Зритель этого фильма не только видел и слышал, но и ощущал своей кожей то, что было записано на пластинке (которую мы сейчас назвали бы магнитным диском).

То, что сделано сейчас, позволяет значительно больше. В фантастическом онирофильме человек лишь воспринимает зрительные, акустические и тактильные раздражители, записанные на носителе — диске. Но он никак не может вмешаться в действие фильма, а является лишь пассивным участником, ощущения которого соответствуют сценарию онирофильма. И если по этому сценарию зрителю (и осязателю) следует поцеловать кого-то в сцене свидания или получить по физиономии в сцене драки, то он получит все сполна независимо от того, хочет он этого или нет.

Современные компьютерные средства позволяют человеку действовать вполне самостоятельно и изучать воображаемый мир, представленный ему компьютером.

Кроме обычного познавательного интереса в такой компьютерной системе ощупывания несуществующего есть и вполне практическая необходимость. Например, рассматривая изометрическое изображение проектируемой детали на экране дисплея, конструктор наверняка захочет увидеть ее в определенном ракурсе или же найти такой ракурс. Для этого надо предоставить ему возможность поворачивать изображение так, как ему захочется, «повертеть» проектируемую деталь в руках, т. е. не только ощупать несуществующую деталь, но изменить ее положение на экране, повернуть ее в нужный ракурс и т. д., причем сделать это собственными руками. Именно такую возможность и обеспечивает компьютерная система.

Как же это реализуется? На руки оператор надевает специальные перчатки, в которых вмонтированы датчики, фиксирующие положения всех суставов кисти, т. е. углы между фалангами пальцев, углы между пальцами и т. д. Это несложные волоконно-оптические устройства, изменяющие при изгибе свое оптическое сопротивление. Так или иначе, но компьютер, получив данные обо всех углах, характеризующих состояние кисти оператора, получает представление о состоянии кисти и может изобразить кисть на экране дисплея. Кроме того, фиксируется положение кистей рук оператора в пространстве. Эти данные о реальных кистях рук оператора вводятся в память компьютера, который и выводит их изображения на экран дисплея.

Если теперь в той же памяти компьютера заложить описание детали, которую предстоит «ощупать», то нетрудно определить, когда и как изображения детали и кистей рук оператора соприкасаются. Точки прикосновения детали к кистям вычисляются компьютером, и в соответствующие места перчаток оператора

посылаются сигналы на тактильные раздражители, которые вмонтированы в перчатки. При возбуждении такого раздражителя он давит на участок кисти (пальцев или ладони), расположенный под ним. Поэтому у оператора и создается впечатление прикосновения к детали. Зная положение точек «прикосновения» кистей оператора к детали, можно определить, как должна поворачиваться деталь под влиянием такого воздействия, и реализовать этот поворот на экране дисплея компьютера.

Такова в общих чертах идея взаимодействия человека с «искусственной реальностью», заложенной в память компьютера. Развивая эту идею, далее нетрудно представить, как можно будет действовать человеку в воображаемом компьютерном мире. Для этого оператору, кроме перчаток, потребуется комбинезон с датчиками положения всех частей его тела и управляемыми раздражителями, которые будут имитировать соприкосновение с миром, заложенным в память компьютера. Экран дисплея должен быть укреплен на голове оператора и его положение контролироваться компьютером. При повороте головы компьютер изменяет изображение на экране дисплея так, чтобы оператор видел именно то изображение предметов искусственного мира, которое он должен видеть при рассматривании такого мира. Этим создается не только эффект полного погружения оператора в искусственную реальность, записанную в памяти компьютера, но и тактильного взаимодействия с ней. При этом взаимодействии оператор может с помощью того же компьютера изменять тот искусственный мир, в котором он действует.

Стоит ли повторять, что для эффективной реализации такого взаимодействия необходимо иметь огромные вычислительные мощности. Такого рада задачи могут решаться только с помощью суперкомпьютеров.

А теперь рассмотрим, как суперкомпьютер может разрешать конфликты, возникающие на тернистом пути человечества.

Компромисс или выбор

Много скандалов было (и, наверное, будет!) в жизни человечества на всех уровнях его существования — от семейного до космического. И каждый имеет одну и ту же структуру — столкновение двух сторон. При этом каждая сторона приводит очень убедительные доводы в пользу своей версии решения проблемы. В такой ситуации обычно переубедить друг друга не удается, и разражается скандал, в котором логика уступает место эмоциям. А это уже никогда не приводит к благополучному разрешению. Именно поэтому спор обычно решается «этажом выше», как говорится «барин рассудит».

Что ж, подход вполне приемлем, если конфликт не затрагивает многих и если судья располагает необходимой информацией о

последствиях каждого способа разрешения конфликта (в пользу одной или другой стороны). Чаще судья старается найти компромисс, чтобы не обидеть ни одну из сторон или обидеть в наименьшей мере. Такое половинчатое решение сохраняет мир, но никак не способствует прогрессу.

Действительно, если одно предложение хорошо, а другое плохо, то их компромисс хотя и будет лучше плохого, но хуже хорошего. Нужно уметь находить лучшее решение. Но принимает решение человек, и «ничто человеческое ему не чуждо». В том числе и ошибки!

А есть ли способ избежать их? Здесь невольно вспоминается старая идея Лейбница — вычислять (без кавычек) истину. Эта идея основана на вере в возможность моделирования человеческих рассуждений. Правда, дальше этого предложения дело не пошло, ведь 300 лет назад еще не было компьютеров. Не пошло бы дело и 30 лет назад, когда компьютеры уже были, но еще малой производительности и с ничтожной памятью. Дело пошло только сейчас, когда появились суперкомпьютеры.

Моделирование рассуждений — очень громоздкая процедура, которая далеко не всякому компьютеру «по зубам». Действительно, чтобы выбрать один из нескольких вариантов решения, нужно прежде всего иметь возможность определить достоверно, к чему приведет выбор того или иного варианта решения. Это означает, что вместо обычных для человека «с одной стороны,..., с другой...» надо иметь инструмент для вычисления всех последствий, инструмент предвидения, как будут разворачиваться события при принятии того или иного решения. Такой инструмент должен состоять по крайней мере из модели обсуждаемого события, на которое воздействуют спорные решения, и вычислительной машины, на которой «обыгрывается» эта модель.

Модель (это понятие также ввел Г. Лейбниц) представляет собой систему утверждений, с помощью которых можно определить поведение некоего явления, не прибегая к самому явлению. Например, второй закон Ньютона — это модель, которая связывает массу, силу, действующую на массу, и ускорение. Именно эта модель позволяет вычислять движение тела как результат приложения силы к этому телу без экспериментирования с ним. Именно модели позволяют достоверно предвидеть, как пойдут события в той или иной ситуации.

Созданием моделей занимается наука, которая является ничем иным, как собранием моделей окружающего нас мира. Эти модели универсальны: они позволяют описывать поведение не одного, а множества явлений одного и того же класса, и достаточно просты, чтобы при их использовании не нужно было слишком много вычислять. Ведь современная наука существует по крайней мере 300 лет (со времен Галилея), а компьютеры, способные ис-

пользовать сложные модели, появились совсем недавно. Поэтому наука всегда ориентировалась на простые модели.

Но жизнь всегда сложнее! И реальные явления, волнующие нас (технические, технологические, экономические, социальные, экологические и т. д.), не удается описывать простыми моделями — уж очень сложны они сами. Именно для решения таких задач был придуман специальный метод, который получил название...

Имитационное моделирование

Имитационная модель отличается тем, что она, во-первых, описывает, как правило, конкретное уникальное явление, а во-вторых, реализуется только на компьютере. Суть имитационного моделирования состоит в том, что все, что происходит в данном явлении, моделируется последовательно во времени и с учетом всех особенностей данного явления и обстоятельств, ему сопутствующих. Например, падение камня с горы можно имитировать, определяя его положение через равные интервалы времени. Для этого нужно знать форму и массу камня, а также рельеф, по которому ему предстоит катиться, и закон всемирного тяготения. Так что для имитационной модели всегда требуется много исходной информации, отражающей специфику моделируемого явления.

Особую роль играет имитационное моделирование в решении экологических проблем. Наша цивилизация овладела настолько мощными средствами изменения окружающего нас мира, что они начинают разрушать Природу. Примеров этому много. Так что же, отказаться от стремления благоустраивать нашу Землю? Отказаться от прогресса? От тех возможностей, которые он дал в руки человечества? Смириться? Ответ напрашивается сам собой. Просто нужно уметь предвидеть экологические последствия своих действий и выбирать то, которое минимально разрушает природу. Конечно, это не просто! Для такого предвидения необходимо создавать экологические модели и с их помощью определять, как влияют на экологию те или иные наши решения.

Экологические модели довольно четко подразделяются на локальные и глобальные. Первые характеризуют локальные изменения в экологической обстановке, например анализ последствий осушения болота, вырубки леса, отстрела дичи, аварии на химическом заводе и т. д. Каждая из таких моделей должна позволять достоверно отвечать на вопрос: «А что будет, если...» и далее следует описание возмущающего фактора, изменяющего экологическую обстановку. Это описание фактически фиксирует начальные условия, в которых начинается функционирование экологической модели. А полученный результат называют *сценарием* —

изложением того, что и в какой последовательности произойдет, если реализуются данные возмущения.

Совершенно аналогично работают глобальные модели. Этот вид моделей отличается тем, что моделируется изменение интересующих нас факторов на всем земном шаре, т. е. реализуются глобальные сценарии. Например, именно глобальным моделированием можно выяснить последствия атомной войны или переброса рек. Как это делается?

Вся поверхность земного шара разбивается на квадраты (точнее, трапеции со сторонами вдоль широт и меридианов). На каждом таком квадрате как на основании строятся кубики (точнее, параллелепипеды) вверх (в атмосферу) и вниз (в землю на суше или в воду на больших реках, озерах, морях и океанах). Каждый такой кубик предполагается однородным — его температура, плотность влажность и т. д. одинаковы по всему объему. Каждый кубик взаимодействует только со своими соседями. Их, как легко подсчитать, шесть. Законы взаимодействия кубиков просты и легко моделируются. Например, разное давление в соседних атмосферных кубиках вызывает перемещение воздуха (обмен массой между ними) со скоростью, пропорциональной разности их давлений.

Как видно, состояние каждого кубика определяется его параметрами, например температурой, массой, давлением, концентрацией интересующих нас веществ (например, CO_2), скоростью движения массы кубика, объемом его биомассы. Зная состояние каждого кубика, всегда можно вычислить, как оно изменится через одну, две, три и т. д. единицы времени. Эту функцию естественно возложить на компьютер.

А теперь подсчитаем загрузку этого компьютера. Она зависит от количества кубиков и числа параметров, определяющих состояние каждого кубика. Очевидно, чем меньше кубиков, тем грубее глобальная модель. Их количество должно быть не менее тысячи (по поверхности Земли), что соответствует очень грубой сетке примерно 1000×1000 км. По вертикали число кубиков должно быть не менее 10, иначе нельзя будет учесть важные процессы вертикального переноса массы (осадки, испарения и т. д.). Итого число кубиков должно быть значительно более 10 тыс. Пусть их будет миллион.

Число параметров, определяющих состояние каждого из этих кубиков, существенно зависит от решаемой задачи. Если это экологические задачи, то необходимо учитывать многочисленные биологические факторы, характеризующие тип и состав биологических сообществ — животных и растений. Число таких параметров доходит до тысячи. Если же решается только климатическая задача, то число параметров несколько меньше.

Итак, глобальная модель характеризуется миллиардом параметров! На каждом шаге все эти параметры изменяются. Для

этого нужно решить миллиард уравнений! Обычный (миллионный) компьютер с этим не справится. Эту задачу может решить только суперкомпьютер, и то не каждый, а «миллиардник». Причем ему придется изрядно потрудиться, чтобы сделать прогноз хотя бы на 1000 этапов смены состояний модели (о меньшем числе не стоит и говорить).

Глобальное моделирование уже позволило получить ряд важных результатов. Так, были рассчитаны сценарии ядерной войны, когда возмущающим фактором явилось выделение большой энергии в ряде кубиков модели, что моделировало нанесение ядерных ударов. Дальнейшее развитие событий всем известно — возникновение гигантских лесных пожаров, дым от которых надолго закроет солнце и в результате ядерная зима, если не выпадет черный дождь. Здесь есть о чем поразмыслить и военным и политикам. Именно такие размышления привели к выводу, что ядерная война является самоубийством человечества.

Вычислительные системы высокой надежности

Существует большой класс практически очень важных задач, при решении которых компьютеру нельзя ошибаться, как, например, при обработке уникальной информации или при управлении реальными процессами (технологическими, управления космическими объектами, не допускающими остановки из-за неполадок в управлении, интенсивной терапии, где малейшая заминка может стоить жизни пациенту, и многие другие). Для решения подобных задач необходимы высоконадежные системы обработки информации. Но каждый элемент вычислительной техники обладает ограниченной надежностью. На первый взгляд кажется, что, собрав вычислительную систему из элементов ограниченной надежности, мы получим систему значительно более низкой надежности, чем ее элементы. Именно так было с первыми ЭВМ — они простаивали и ремонтировались значительно дольше, чем работали, так как состояли из огромного числа ненадежных элементов. Так продолжалось до тех пор, пока вопросам надежности не стали уделять специальное внимание и создавать специальные архитектуры вычислительных систем повышенной надежности. Это достигается введением все той же параллельности, как и при создании высокопроизводительных вычислительных систем.

Одним из простейших принципов повышения надежности является дублирование (или троирование) работы системы. Возьмем, к примеру, три одинаковые ненадежные ЭВМ и введем в них одну и ту же программу. Кроме этого, организуем оперативное сопоставление результатов работы этих ЭВМ через определенные интервалы времени. Если все три результата совпадают, то естественно считать, что эти ЭВМ работают правильно (случай, когда все они ошиблись одинаково при решении одной и той же

задачи, исключаем ввиду его крайне малой вероятности). Если же появляется разнобой в полученных результатах, то, очевидно, какой-то из компьютеров ошибается. Как же быть в этом случае? Довольно просто — надо голосовать! При этом каждый компьютер голосует за свой результат и несложное устройство выбирает тот результат, который собрал максимальное число голосов. Если ошибся один компьютер, то все равно будет принято правильное решение, так как два компьютера дали одинаковый результат, который, набрав два очка против одного, и будет выдан как решение тройки компьютеров. (Возможна и одинаковая ошибка двух компьютеров, и тогда будет принято ошибочное решение. Но вероятность такой ситуации крайне мала и может смело не приниматься в расчет.) И только когда два компьютера ошибутся одновременно (и по-разному), голосующее устройство будет поставлено в тупик — ведь все три компьютера выдали разный результат. Но такое событие происходит очень редко. Несложные расчеты показывают, что оно произойдет один раз в 100 тыс. лет, если каждый из трех компьютеров ошибается один раз за три часа своей работы.

Это при случайном сбое работы компьютера, т. е. при однократной ошибке. Если же в такой «троированной» вычислительной системе откажет один компьютер (перестанет правильно работать), то два остальных смогут некоторое время поддерживать работоспособность системы (на время ремонта вышедшего из строя компьютера). Но при этом уже сбой в работе одного из двух работающих компьютеров будет лишь фиксироваться (по несовпадению результатов), что можно использовать, например, для повторения тех вычислений, в процессе которых произошел сбой. На это есть веские основания, ведь сбой случайный и повторяется нечасто. Поэтому стратегия повторения ошибочных расчетов вполне себя оправдывает и позволяет преодолеть неприятности, вызываемые случайными сбоями в компьютере.

Легко заметить, что надежность такого комплекса из трех машин значительно превышает надежность одной машины. Но эта схема слишком расточительна — она требует утроения всего вычислительного ресурса.

Можно ограничиться удвоением, если в каждую машину встроить устройства, контролирующие правильность работы элементов ЭВМ. Такой контроль позволяет сразу отключать отказавшую ЭВМ и сигнализировать обслуживающему персоналу, какой блок отказал, чтобы его заменить на новый или сделать это автоматически. Для этого каждый блок ЭВМ следует дублировать параллельно работающим точно таким же блоком и иметь подсистему контроля правильности его работы. Тогда, снимая информацию лишь с правильно работающих элементов, получаем безостановочную систему обработки информации. Действительно, при отказе одного из элементов системы сигнал о его неисправности

поступает оператору, и замена этого элемента на новый происходит без остановки системы — его функции выполняет дублер.

Всякое электронное устройство имеет очень важную характеристику — наработку на отказ (среднее число часов безотказной работы этого устройства). И только задублированные описанным образом устройства не имеют такой характеристики — они никогда не отказывают. Примером такого рода компьютера, все электронные блоки которого задублированы, является наша ЭВМ ЕС-1045. Это, естественно, не означает, что такой компьютер работает совсем безотказно. Кроме электронных блоков, всякая ЭВМ имеет механические устройства (дисководы, клавиатура пульта и т. д.), которые нельзя или очень трудно задублировать и тем более автоматически выявить их неисправность. Именно они снижают надежность компьютера.

Таким образом, вычислительные системы для самых разнообразных целей (повышенной производительности, высокой надежности и т. д.) создаются с использованием идеи распараллеливания работы различных элементов, блоков и устройств вычислительных систем. Такая параллельная обработка позволяет получать необыкновенно высокие характеристики современных вычислительных систем, что является новым шагом в развитии вычислительной техники.

— Давно известно, — заметил Мегрэ, — что развитие передовых областей техники в нашем мире стимулируется военными. Это понятно, у военных всегда было много средств и приоритетов, они и «заказывали музыку», точнее передовую технику. А гражданские пользовались тем, что падало с марсова стола. Так было, есть и, наверное, будет. Неужели и компьютерная техника не исключение?

— К сожалению, нет, — вздохнул Поль. — Вы помните, первый компьютер был создан в США для расчета баллистических таблиц. Именно военные приложили руку к развитию вычислительных машин. Но, к счастью, возможности компьютеров в мирной области оказались столь впечатляющими, что развитие их надолго определилось интересами науки, экономики, техники и т. д.

— Так что закон нарушен, и компьютеры восторжествовали над военными? — изумился Мегрэ.

— Да, нет, — грустно сказал Поль, — довольно скоро военные поняли, что компьютеры могут применяться не только для проектирования военной техники. А военные применения компьютера требовали от него сверхвысокой производительности, огромной памяти и надежной коммуникации с источниками информации и исполнителями. Именно в этом направлении и пошло развитие вычислительной техники. Но этого же от нее ждали и мирные области.

Наиболее интересна программа КОИ (Компьютерная Оборонная Инициатива) США, принятая в 1984 г. В ней до 1993 г. планируется, например, создать три вида систем. Первая — автономное транспортное средство для разведки, снабжения войск и доставки боеприпасов на поле боя. Это целиком автоматизированное самоходное средство, снабженное мощным компьютером для планирования действий исходя из обстановки. При скорости 10—20 км/ч для выполнения поставлен-

ных целей (доставки грузов по пересеченной местности в заданную точку) компьютер должен иметь производительность 100 млрд. операций в секунду и емкость памяти в 10—100 Гбайт, т. е. 10—100 млрд. байт. Конечно же, это будет многопроцессорная компьютерная система с широким распараллеливанием процесса обработки поступающей с поля боя информации.

— Это же транспортный робот, о котором давно мечтают и в сельском хозяйстве, и на производстве! — воскликнул Мегрэ. — Его почти без переделки можно будет использовать.

— Разумеется, и такая преемственность предусмотрена в КОИ, — заметил Поль. — Другая система — помощник пилота-истребителя, на которого возлагаются ведение самолета, распознавание целей, ведение боя, информирование летчика о состоянии важнейших агрегатов самолета, обработка данных, получаемых с радиолокационных станций, и т. д. Причем с летчиком эта система должна общаться на естественном языке в условиях больших ускорений, шумов и вибраций. Летчик может запросить голосом любые данные о состоянии самолета, обстановке в воздухе, попросить совета и т. д. Ответ он должен получить немедленно. Стоит ли говорить, что для создания такого помощника необходимо иметь компьютерную систему гигантской производительности — порядка 10^{13} операций в секунду, т. е. десять тысяч миллиардов (!), и ... малых габаритов. И, наконец, система управления боевыми действиями кораблей военно-морских сил совместно с авиацией (ее компьютеры будут расположены на каждом боевом средстве) должна обеспечить представление картины боя на дисплей в центре управления, управлять им, предлагать направления наносимых ударов, прогнозировать погоду и давать прогноз на четверо суток при запросе «а что будет, если...». Всю эту огромную работу должна выполнить компьютерная система огромной производительности.

Интересно, что основные вычислительные затраты здесь идут на подсистему видения, обработки видеоинформации, поступающей в компьютер с поля боя. Поэтому компьютерная система, активно взаимодействующая с окружающим миром через подсистему видения, должна иметь огромные вычислительные мощности для обработки видеоинформации.

— Все это очень интересно, — грустно заметил Мегрэ, — но, к сожалению, так далеко от наших с вами профессиональных интересов и забот. Ведь общаться с компьютером нам приходится через клавишный пульт. А это очень узкий канал передачи информации компьютеру. Если бы, как в этих проектах, компьютер мог видеть, то ему можно было бы давать более серьезные поручения. Ведь поговорка «Лучше один раз увидеть, чем десять раз услышать» относится и к компьютерам.

— Не тужите, шеф, — успокоил его Поль. — Такой «зрячий» компьютер вы получите наверняка. Но после того, как он будет рассекречен военными, — такова наша жизнь. Это еще не самый худший случай, когда военные передают свои разработки штатским. Волей-неволей им приходится таким образом двигать прогресс. А когда оборона не понадобится (ведь настанет же такое время?) прогресс вычислительной техники будет происходить в соответствии с нашими требованиями. Я уверен, что это не за горами.

— Вы оптимист, Поль. Но хочется надеяться, что так и будет, настанет время сыщика с компьютером (а точнее, с компьютерной системой).

7. АЛЛО, ОТВЕТЬТЕ КОМПЬЮТЕРУ!
(Вычислительные сети)

СРЕДСТВО ПЕРЕРАСПРЕДЕЛЕНИЯ РЕСУРСОВ

Всякий, кто овладел компьютером (любым — персональным или коллективного пользования), вскоре начинает думать, как бы расширить его возможности. И дело тут вовсе не в капризах пользователя, а в ограниченных ресурсах любого компьютера. Во-первых, хотелось бы повысить производительность «своего» компьютера за счет других, которые в это время простаивают, и, во-вторых (самое главное), получить доступ к их памяти, где хранятся интересующие пользователя данные, и прежде всего программы. Если первое желание связано с вполне естественной неудовлетворенностью, которая всегда наступает вскоре после овладения любым компьютером, то второе нуждается в обосновании. Дело в том, что процесс общения со всяким компьютером сводится в конечном счете к общению с его программами. И чем больше таких программ, тем эффективнее работа компьютера. Без большого преувеличения можно сказать, что интеллектуальность компьютера пропорциональна числу его программ. Именно поэтому всякий пользователь заинтересован в расширении своего информационного ресурса, в получении большого числа разнообразных программ.

Итак, именно ограниченность ресурсов компьютера (памяти, вычислительной мощности, программного обеспечения, необходимых данных и т. д.) заставляет создавать вычислительные сети, которые позволяют почти неограниченно расширять ресурсы, необходимые пользователю. Это и определяет неослабный и устойчивый интерес, проявляемый к вычислительным сетям в последние годы.

Всякая вычислительная сеть, вообще говоря, образуется только двумя подсистемами: сетью связи и вычислительными машинами, подключенными к ней. Здесь абонентами сети связи являются компьютеры. Причем та часть, за которыми работают пользователи, называют абонентскими машинами, а остальные — главными ЭВМ или ГЭВМ. Именно ГЭВМ являются источниками ресурсов вычислительной сети, предоставляемых их пользователям.

С помощью сети связи производится передача ресурсов от одной машины к другой. Так как это надо делать быстро и надежно, то в сети связи также используются компьютеры, но специального назначения — для быстрой обработки передаваемой по сети информации. Одной из важнейших функций такой обработки является коммутация информации по различным направлениям (аналогично автоматическим телефонным станциям в сети телефонной связи). Эту функцию в вычислительной сети выполняют узлы коммутации, построенные на коммутационных машинах (обычно это мини-ЭВМ). Именно коммутационные машины обеспечи-

вают эффективность функционирования сети связи или, как ее еще называют, коммутационной сети. Очевидно, что обмен ресурсами в вычислительной сети происходит очень быстро — со скоростью коммутации информации в сети связи. Но такое объединение средств связи с вычислительной техникой не дается даром и требует разработки специальных подсистем взаимодействия компьютеров с каналами связи.

ИНТЕРФЕЙСЫ И ПРОТОКОЛЫ

Взаимодействие ЭВМ и сети связи осуществляется через интерфейс (напомним, что интерфейс — это средство сопряжения оборудования, программ и т. д., т. е. граница, через которую осуществляется взаимодействие между двумя контактирующими подсистемами). Информация, проходящая через интерфейс, должна передаваться в определенной форме — в виде сигналов заданной длительности, уровня, формы, последовательности и т. д. Примером использования самого простого интерфейса является подключение к электрической сети бытовых приборов. Здесь интерфейсом является пара вилка-розетка, а требованием интерфейса — необходимость соответствия вилки и розетки. (Вспомним, что интерфейсы, т. е. вилки-розетки электрической, радио- и телефонной сетей различны, чтобы их нельзя было перепутать.)

Всякое отклонение от требований интерфейса приводит к нарушению взаимодействия смежных подсистем. Требования интерфейса аналогичны языковым требованиям при общении людей: чтобы понять друг друга, они должны говорить на одном языке, т. е. их интерфейс должен выполнять роль переводчика с одного языка на другой. Эта важная функция интерфейса часто превращает его в самостоятельную подсистему. В качестве интерфейса может выступать, например, ЭВМ, которую в этом случае называют *интерфейсной.* На нее возлагается функция сопряжения абонентской машины с коммуникационной сетью.

Но для функционирования вычислительной сети одного описания границ ее подсистем недостаточно. Надо еще определить правила взаимодействия удаленных подсистем сети (например, двух ЭВМ, подключенных к разным концам вычислительной сети), не имеющих общего интерфейса. Такие правила не нужны в многопроцессорных и многомашинных вычислительных системах. Все подсистемы этих систем подчинены одной цели, которая достигается выбором соответствующей единой архитектуры и необходимых интерфейсов. Система образуется одинаковыми процессорами или машинами. Вычислительная же сеть состоит из разных ЭВМ, их цели разные, на них решают свои задачи разные пользователи. Именно поэтому должны быть строго определены правила взаимодействия различных подсистем вычислительной сети. Например, как вести диалог через сеть, чья очередь

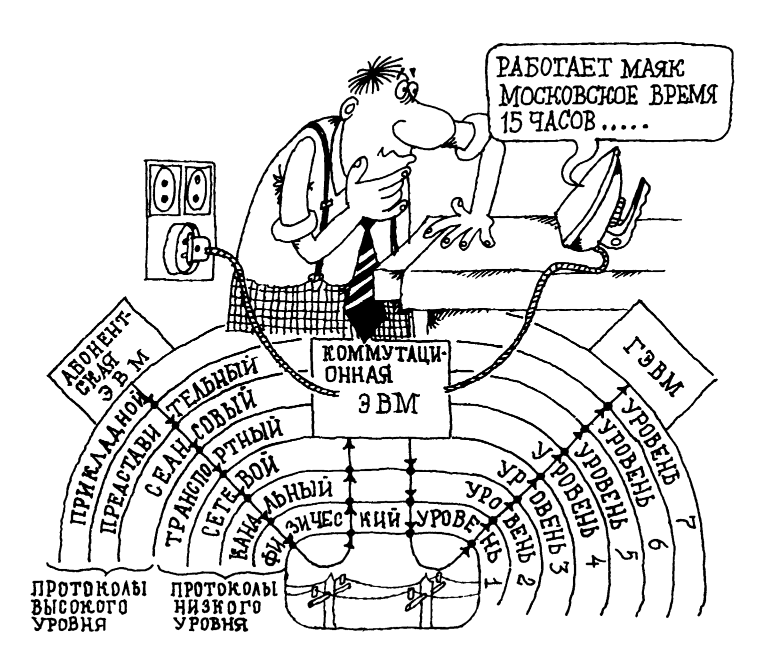


Рис. 14

передавать информацию, что нужно делать, если сообщение не принято, и т. д.

Эти правила называют *протоколами*. В протоколах вычислительной сети строго регламентированы правила взаимодействия удаленных подсистем сети в различных ситуациях. В соответствии с этими протоколами и производится преобразование информации при ее движении по сети. В процессе этого движения информация проходит различные уровни, функционирование которых определяется соответствующими протоколами. Эти уровни напоминают луковицу — в центре средства связи, а пользователи общаются с ней через «наружную кожуру» (рис. 14). Каждый слой такой «луковицы» образует уровень, работа которого регламентируется необходимыми протоколами, а связь между уровнями — соответствующими интерфейсами (точки на рис. 14). Информация от одной ЭВМ сети к другой всегда проходит все уровни в том и другом направлении, изменяя свою форму в соответствии с протоколами каждого уровня. Всего по международным рекомендациям установлено семь таких уровней. Это вовсе не означает, что каждая вычислительная сеть обязательно должна иметь все семь уровней. Их может быть и меньше, но тогда все семь функций будут перераспределены между этими уровнями.

Проиллюстрируем работу протоколов и интерфейсов на примере бюрократического взаимодействия двух организаций, неважно каких (здесь слово «бюрократия» используется в своем изначальном смысле как соблюдение формаль-

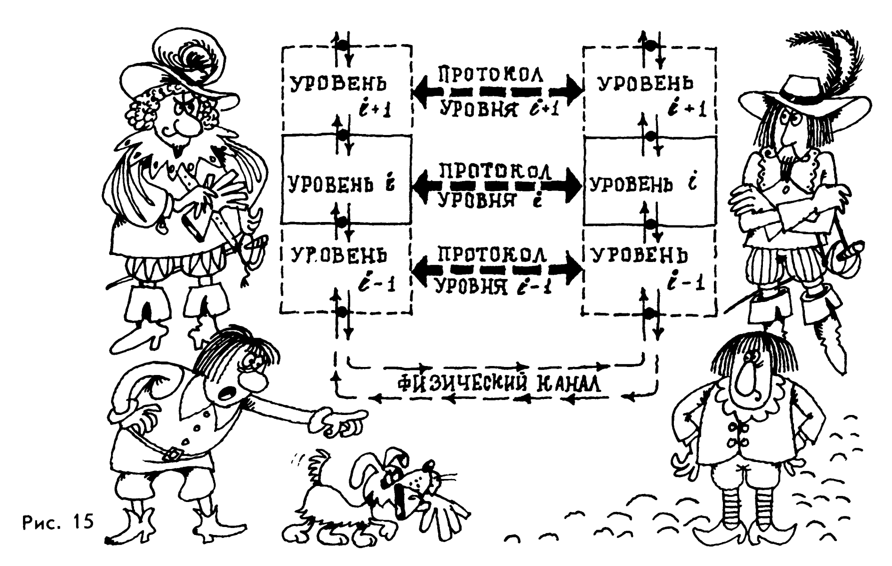


Рис. 15

ных правил). Пусть директор одной организации решил направить сообщение директору другой. Это сообщение должно пройти несколько уровней обработки — директор, его секретарь, канцелярия, почтовое отделение, транспортная система, которая доставит сообщение в почтовое отделение адресата. Далее то же, но в обратном порядке: почтовое отделение, канцелярия, секретарь, директор, которому направлено сообщение. Все эти уровни взаимодействуют между собой по правилам общения непосредственно контактирующих подсистем, т. е. по правилам интерфейса. Нарушение этих правил приводит к нарушению взаимодействия. Если, например, директор, передавая письмо своему секретарю, укажет не официальное имя адресата, а его кличку среди коллег, то секретарь его не поймет и не будет знать, куда отправлять письмо. Аналогичны интерфейсы и других уровней.

Таким образом, каждый уровень имеет два интерфейса — на входе в этот уровень и на выходе из него. Причем каждый интерфейс работает в обе стороны, обеспечивая передачу и получение информации как «сверху», так и «снизу» (рис. 15). Как видно, интерфейсы здесь обеспечивают взаимодействие уровней по «вертикали». Кроме этого, каждый уровень обработки информации неизбежно взаимодействует еще и со своим уровнем адресата, т. е. по «горизонтали». Именно эти взаимодействия происходят по протоколам (рис. 15).

Так, директор обращается с письмом к директору (а не к его секретарю), и это общение происходит по протоколу корректных деловых взаимоотношений. Если он не будет соблюдать этого протокола, например начнет командовать, чем нарушит установленный протокол общения между директорами, то его коллега попросту не ответит на письмо, и взаимодействия директорских уровней не произойдет. Аналогично секретарь должен позаботиться, чтобы его коллега у адресата подготовил все необходимое для ответа на полученное письмо. По протоколу этого уровня необходимо указать повод, по которому написано это письмо, т. е. сослаться на соответствующие документы и обстоятельства, которые

заставили написать это письмо. Получив эти данные, секретарь адресата будет знать, какие именно материалы нужно подготовить для директора к этому письму. Аналогично взаимодействуют и другие уровни.

Как видно, разница между интерфейсом и протоколом заключается в том, что интерфейс определяет правила непосредственного взаимодействия подсистем, которые реализуются в виде схем или программ, а протокол определяет правила взаимодействия удаленных подсистем (точнее, их уровней), которые реализуются в виде процедур взаимодействия.

Одной из важнейших функций протокола является обеспечение независимости взаимодействия уровней. Действительно, директора могут договориться изменить протокол своего взаимодействия и обращаться друг к другу, например, не официально, а по-дружески, фамильярно. Изменение этого протокола никак не скажется на протоколах других уровней — они действуют независимо. Это обеспечивает гибкость системы передачи информации, необходимую для всякой сложной системы, которая может изменяться в процессе своего функционирования. Например, адаптируясь к состоянию среды, в которой она действует.

РЕСУРСЫ, РЕСУРСЫ, РЕСУРСЫ...

Рассмотрим их перераспределение с помощью передачи сообщений через сеть от одного компьютера к другому. Такое взаимодействие компьютеров через сеть является универсальным средством передачи ресурсов. Для увеличения производительности своего компьютера достаточно иметь возможность передавать свои программы для выполнения на другом компьютере. Это можно сделать, послав сообщение в виде программы и исходных данных на другой компьютер (ГВМ — главную вычислительную машину), который, решив задачу, т. е. выполнив программу, отправит полученный результат обратно. Вы же при этом можете на своем компьютере решать другие задачи. Это не что иное, как повышение производительности вашего компьютера.

Другой вид ресурса связан с необходимостью получения каких-то сведений или программ из удаленного банка данных. Этот информационный ресурс вы можете получить через сеть, если ваш компьютер подключен к ней. В ответ на ваш запрос (это тоже сообщение) ГВМ, передает вам сообщение, содержащее необходимую информацию. И, наконец, желая разместить в памяти компьютера большой объем информации, например текст книги, но не располагая таким объемом памяти, можно этот объем передать по сети в виде сообщения в ГВМ, имеющую свободную память для хранения. Эта информация будет возвращена вам немедленно

при обращении к этой ГВМ. Это и есть расширение ресурса памяти вашего компьютера.

Как видно, перераспределение ресурсов сети происходит только путем передачи сообщений. Правила преобразований и движений сообщений по вычислительной сети регламентируются протоколами взаимодействия ее подсистем.

Коммуникация в нашем мире и благо и ... бедствие — вместе с огромной пользой она несет в себе и большие неприятности. На уровне человеческого общения коммуникация приводит к распространению вирусных заболеваний по всему земному шару. Это и грипп, с которым удается справляться, и СПИД, с которым справиться пока не удалось. В системе информационной коммуникации появились компьютерные «вирусы», которые распространяются по каналам связи и причиняют ущерб не меньший, чем пандемии гриппа и СПИДа.

Известно, что самым ценным в любой компьютерной системе является ее программное обеспечение. Оно раз в десять дороже компьютерного «железа». Но именно программное обеспечение наиболее уязвимо. Эта уязвимость и соблазнила некоторых программистов создать программы-вирусы.

Программа-вирус обычно очень компактна и может воспроизводить себя в разных частях памяти компьютера. Вместе с передаваемыми по вычислительной сети сообщениями распространяется и программа-вирус, разрушая содержимое памяти всех компьютеров, взаимодействующих через сеть связи. Ущерб, наносимый одним таким вирусом, исчисляется огромными цифрами, не говоря уже о моральных потерях.

Как же устроена эта программа-вирус? Едва ли стоит об этом говорить подробно, чтобы не соблазнить безответственных программистов. Приведем лишь один пример.

Есть такой язык низкого уровня (символический) — Редкод (Redcod). Он содержит лишь 10 простых команд и используется обычно для составления программ компьютерных игр. Одна из команд этого языка MOV предназначена для перемещения информации в памяти. Она состоит из трех частей — кода команды и двух адресов — и занимает одну ячейку памяти:

$$\text{MOV}\ A\ B,$$

где A и B — эти адреса. Если такая команда расположена в ячейке с номером (адресом) C, то при ее выполнении содержимое ячейки с адресом $C+A$ переместится в ячейку с адресом $C+B$. Это очень удобная и полезная команда. Но при $A=0$ и $B=1$ она становится тем самым вирусом, который начинает размножаться и уничтожать подряд содержимое памяти компьютера. Действительно, обращение к ячейке с адресом $C+0=C$ дает сам оператор MOV 0 1, который в соответствии с его смыслом (семантикой) будет записан в ячейку с адресом $C+1$, т. е. в соседнюю. А так как команды языка Редкод выполняются последовательно (как

в Фортране и Алголе), то компьютер будет выполнять следующую команду, расположенную в ячейке $C+1$, т. е. запишет MOV 0 1 в ячейку $C+2$, и т. д. В результате эта команда будет перемещаться от адреса к адресу по всей памяти компьютера, сокрушая все на своем пути и оставляя за собой опустошающий след из команд MOV 0 1.

Это лишь пример простейшего компьютерного вируса, состоящего из одной команды. С ним легко справиться. Но разработчики программ-вирусов обеспечивают им защиту. К сожалению, им это удается. И тогда содержимое памяти всех связанных сетью компьютеров превращается в электронное конфетти. Такая программа-вирус становится логической бомбой. При наличии коммуникационной сети логическая бомба «взрывается» в каждом компьютере этой сети, рушит всю сеть, которая может быть распределена по всей Земле.

Заметим, что оборонные службы всех развитых стран серьезно озабочены реальностью такой компьютерной диверсии, разрушающей сеть системы обороны, и изыскивают все возможные средства, чтобы не допустить попадания логических бомб в оборонные сети.

Если заранее и наверняка знать, что в память вашего компьютера попала логическая бомба, то выявить ее не представляет труда по неадекватной реакции на те или иные тестовые программы, но только в том случае, если бомба не взорвалась. Именно поэтому иногда ее делают замедленного действия. Она «взрывается» лишь после определенного числа обращений компьютера к программе этой бомбы, чем очень затрудняется ее выявление.

Началось все с простого компьютерного хулиганства. А при отсутствии элементарной этики и при наличии мощных средств коммуникации в современных компьютерных сетях проблема компьютерной чумы, вызванной программой-вирусом, превращается в общечеловеческое бедствие, когда уничтожаются огромные ресурсы информации. Эта проблема, как и проблема СПИДа, ждет своего решения.

В компьютерных системах, даже самых простых, всегда есть защита от несанкционированного доступа. Это значит, что какая-то определенная часть памяти защищена от произвольного вторжения извне и доступ к ней осуществляется по специальному ключу (паролю), известному лишь владельцу этой части памяти. Делается это не только против злоумышленников, но и просто для защиты важной информации от случайных воздействий. Например, от ошибок в программах. Так, например, защищают ту часть программы, где расположена операционная система компьютера, так как нарушение ее работы делает невозможным его функционирование. Стоит ли говорить, что мощная защита преграждает доступ к секретной информации — финансовой, военной, оборонной и т. д.

Эта защита создается так, чтобы никто не смог подобрать ключ к ней — об этом заботятся программисты экстракласса. Но ... созданные трудности вызвали желание их преодолеть, и на Западе среди программистов появились так называемые хэккеры — «взломщики» программных защит. Такой хэккер, сидя дома за своим компьютером, подключенным к вычислительной сети, пытается подобрать программную «отмычку» к замкам, расставленным на компьютерах, подключенных к этой сети. Делает он это не всегда бескорыстно. Так, подключившись к файлам, хранящим счета в банках, он может перевести на свой счет любую сумму с других счетов. Причем поймать его «за руку» при однократном подключении почти невозможно, ведь банки хранят счета только в компьютерной памяти. Это, пожалуй, самая «безобидная» пакость такого хэккера. В принципе он может спровоцировать даже атомную войну, если решение о ней принимается или передается компьютером, подключенным к сети. А так как хэккеров много и работают они не жалея времени, то и ущерб, наносимый ими, огромен. Так в юриспруденции появилось новое понятие «компьютерное преступление» — преступление, совершенное с помощью компьютера.

Но хватит о негативных сторонах компьютерных сетей — виноваты в этом не сети, а злая воля компьютерных хулиганов. Поговорим о системе протоколов, позволяющей сети эффективно перераспределять ее ресурсы.

СЕМЬ УРОВНЕЙ ПРОТОКОЛОВ

Прежде всего следует отметить, что все протоколы взаимодействия в сети подразделяются на протоколы высокого (это уровни от 5 до 7) и низкого (от 1 до 4) уровня (рис. 14). Отличаются они тем, что верхние уровни учитывают специфику передаваемой информации, а нижние — индифферентны к ней, для них это лишь поток нулей и единиц. Аналогичное подразделение протоколов имеет место в телефонной связи, где верхний уровень протоколов требует соблюдения общепринятого человеческого этикета — говорить вежливо, отвечать на вопросы собеседника, не перебивать его и т. д. Если вы нарушите эти протокольные правила, то ваш собеседник вправе повесить трубку, прервать связь. Нижний же уровень протоколов телефонной сети обеспечивает технически связь между абонентами, т. е. обмен сигналами в определенном интервале частот. Что именно будет передаваться (музыка, речь или «морзянка»), для этого протокола неважно.

Протоколы нижних уровней реализуются в сети связи (ее еще называют коммуникационной или транспортной сетью), а протоколы верхних уровней — вне ее. Таким образом, граница сети связи определяется интерфейсом верхних и нижних уровней протоколов.

Итак, сформированное сообщение, пройдя три верхних уровня (прикладной, представительный и сеансовый), необходимые для определения формы взаимодействия (на прикладном уровне), представления и преобразования сообщения в определенную унифицированную для данной сети форму (на представительном уровне) и установления сеанса связи между абонентами (на сеансовом уровне), попадает, наконец, в транспортную сеть (сеть связи).

Транспортная сеть обеспечивает транспортировку вверенных ей «грузов»— информации пользователя. Ее задача быстро и надежно доставлять эти «грузы» адресату. Быстрота доставки обеспечивается электроникой, а надежность — специальными способами защиты передаваемой информации от воздействия внешних и внутренних помех. Так же, как при транспортировке, грузы нуждаются в защите от ударов, передаваемая информация защищается от «ударов» — помех.

Поступившее сообщение, прежде чем быть отправленным по каналу связи, подвергается обработке. Эта обработка нужна, чтобы обеспечить надежную передачу сообщения по сети. Дело в том, что при движении по сети информация подвергается довольно интенсивной «бомбардировке» помех. И достаточно помехе исказить лишь один бит сообщения (т. е. поменять 1 на 0 или 0 на 1) или потерять его, то все сообщение сразу обесценивается. Чем длинее это сообщение, тем оно более уязвимо для помех. Опыт показывает, что сообщение длиной в 10 тыс. бит (это примерно 1000 байт, т. е. 1000 знаков) будет почти всегда передано с ошибкой. А ведь это очень короткое сообщение — меньше одной страницы текста. Нужно же передавать тысячи страниц. Как же быть? Нужна защита от помех. Для этого громоздкое сообщение дробится на мелкие пакеты (так обычно называют самую мелкую долю транспортируемой информации), и эти пакеты «россыпью» запускают в сеть связи. Такова основная идея. А реализуется она следующим образом (рис. 16).

Сначала (на транспортном (4-м) уровне) сообщение дробится на фрагменты одинакового размера (аналог дробления груза для контейнерной перевозки), причем каждому фрагменту добавляется заголовок передачи, где указываются номера сообщения и фрагмента, а также имя получателя. В таком виде фрагмент с заголовком называют обычно блоком. На приемном конце, на том же 4-м уровне, получив этот блок, по заголовку определяют его номер, составляют (восстанавливают) из фрагментов переданное сообщение и отправляют его на верхние уровни, где оно преобразуется к исходному виду, доступному для понимания адресата — пользователя или его компьютера.

Далее (на 3-м уровне) образуется пакет путем добавления к блоку заголовка, где указываются все необходимые сведения о маршруте в сети связи, по которому должен пройти этот пакет.

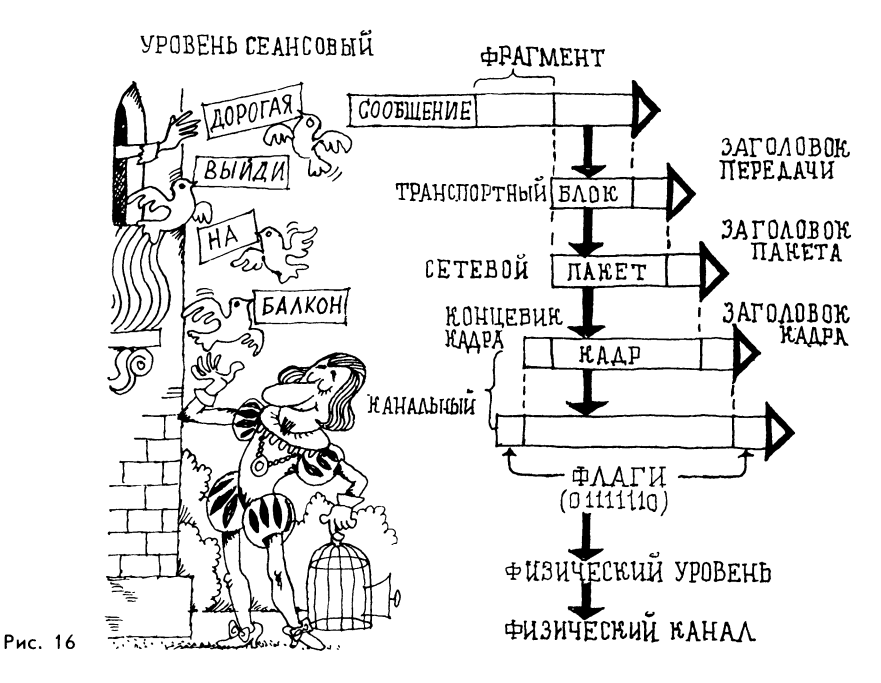


Рис. 16

На канальном (2-м) уровне пакет преобразуется в кадр. Для этого он получает заголовок и концевик. В заголовке передается информация, необходимая для управления каналом связи, по которому предстоит пройти этому кадру. Такой информацией может быть, например, подтверждение правильного приема предыдущего кадра. Концевик кадра предназначен для защиты системы передачи от воздействия помех. Обычно он содержит 2 байта (16 бит) контрольной информации, которая позволяет определить, правильно или неправильно передан этот кадр.

Так трансформируется сообщение, проходящее через транспортную сеть. Все эти «протокольные» преобразования направлены на выполнение одной цели — надежной передачи сообщения. На приемном конце все протокольные операции делаются в обратном порядке, с тем чтобы восстановить переданное сообщение.

ЗАЩИТА ОТ ПОМЕХ

Всякая защита информации от помех начинается с выявления ошибки, появившейся в результате воздействия помех на канал передачи информации. Нет каналов без помех — они неизбежны. И нужно лишь уметь нейтрализовать их разрушительное действие. Именно для этого передаваемая информация сопровождается контрольными битами.

Идея контрольных битов состоит в том, что они определяют-

ся однозначно передаваемой информацией. Например, проверка на четность позволяет одним конрольным битом проверять четность или нечетность переданной информации. Для этого достаточно определить, четно или нечетно число единиц в кадре, и в первом случае в контрольный бит поставить нуль, а во втором — единицу. Если в результате помехи исказится один бит, т. е. 1 станет 0 или наоборот, то четность изменится, что легко обнаружить при приеме кадра.

Описанная проверка на четность лишь иллюстрирует идею защиты передаваемой информации (точнее, это еще не защита, а лишь возможность выявления ошибочной передачи). В действительности контрольных битов 16, и они определяются по передаваемой информации с помощью специального устройства. На приемном конце аналогичное устройство проверяет соответствие принятых контрольных битов и полученной информации. При отсутствии такого соответствия фиксируется ошибочность передачи кадра.

Такая возможность определить, правильно или неправильно был передан кадр по каналу связи, гарантирует защиту от помех, если при этом дать возможность принимающему узлу переспрашивать передающий. Так же, как при телефонном разговоре с помехами, мы просим повторить сказанные нашим собеседником отдельные слова и фразы, система связи с переспросом позволяет принимающему узлу «переспрашивать» неправильно принятый кадр. Для этого кадр должен храниться в памяти передающего узла до тех пор, пока приемный узел не подтвердит правильность приема этого пакета. И лишь получив такое подтверждение, передающий узел может начать передачу следующего кадра и «забыть» предыдущий.

Это означает, что приемный узел, получив пакет, в любом случае должен отправить передающему узлу подтверждение в виде так называемой квитанции. Эта квитанция направляется с очередным кадром встречного потока, идущего от приемного узла к передающему. И располагается эта квитанция в заголовке кадра, который служит для передачи служебной информации. Если же встречного потока кадров нет, то для квитанции организуется специальный служебный кадр (точнее, это «кадрик», очень короткий, так как не несет пользовательской информации). Если квитанция положительная, то, получив ее, передающий узел направляет приемному очередной кадр и забывает предыдущий (точнее, его место в ОЗУ сетевого процессора занимает очередной кадр). Если же он получает отрицательную квитанцию, то снова отсылает кадр, хранящийся в его памяти.

Как видно, помехи в такой системе с переспросом не искажают, а лишь несколько задерживают движение кадров по вычислительной сети.

Здесь Мегрэ недовольно хмыкнул, вспомнив свое армейское прошлое. Он-то хорошо понимал, как важно знать, что отданные распоряжения правильно поняты подчиненным.

— Поль! А почему бы и в вычислительных сетях не воспользоваться этим проверенным правилом? Получив сообщение, приемник запоминает его и отправляет обратно в том же виде. А передатчик сравнивает его с хранящимся в памяти (он перед отправкой сообщения приемнику запомнил его) и, если оно совпало, отправляет положительную квитанцию приемнику (это команда «выполняйте»), подтверждающую, что тот понял правильно, и начинает передачу следующего кадра. Если же совпадения нет, то нужно снова повторить передачу того же сообщения, Именно так поступает командир, отдавая свои распоряжения.

— Так сделать можно, но такая система передачи будет хуже, чем описанная в книге.

— Это почему же?— обиделся Мегрэ.

— Да потому, что в этом случае сообщение передается дважды, что вдвое повышает вероятность его искажения Не говоря уже о том, что на такую связь нужно вдвое больше времени. Вот и получается, что «ваш армейский» способ передачи сообщения вдвое и хуже (ошибочней), и дольше.

— Но если есть более эффективный способ, то почему бы его не использовать в армии?

— Нет,— улыбнулся Поль,— это способ общения машин, а не людей. Если все-таки реализовать его в армии, то командир должен сопровождать приказ контрольной фразой, в которой закодирован смысл этого приказа. Подчиненный, получив приказ и контрольную фразу, сравнивает их и, не обнаружив расхождения, говорит «слушаюсь». Если же они противоречат, то переспрашивает начальника.

— А как строится эта контрольная фраза?

— Например, «В приказе число гласных 75, а согласных 87» или как-то иначе, но в таком же роде.

— Да-а-а,— протянул, улыбнувшись Мегрэ,— не хотел бы я оказаться на месте подчиненного..., да и начальника тоже. Общаться им придется с калькуляторами в руках. Этот способ действительно подходит лишь для машин.

— И все-таки он используется людьми,— лукаво заметил Поль.— Но в «очеловеченном» виде. Для этого нужно контрольную фразу построить в виде резюме, в котором перечисляются главные положения сообщения (резюме иногда называют аннотацией или рефератом). В резюме дублируется смысл сообщения, что и позволяет приемнику выявить правильность понимания сообщения (и его резюме) и либо исполнить его, либо переспросить.

— Сержант Поль! — повысил голос Мегрэ, и Поль встал и подтянулся. — Принимайте сообщение. Не морочьте мне голову наукообразными фразами. Ничего по сути нового для повышения надежности общения ваша компьютерная премудрость не придумала. Все то же повторение или переспрос. А контрольные биты в кадре — это по сути его резюме, но составленное компьютерными средствами. Мое резюме: все это по идее было раньше, а при реализации в вычислительной сети используются вычислительные возможности компьютера. Не так ли, Поль?

— Вы абсолютно правы, шеф, но эти самые «возможности» компьютера настолько велики, что позволяют создать систему эффективного общения между

огромным числом компьютеров в вычислительной сети. Человеческого аналога такого общения просто не существует. Ведь в самом организованном коллективе людей нет возможности одновременно каждому говорить с каждым. В вычислительной сети это возможно. И эту возможность обеспечила «компьютерность коллектива» ЭВМ.

— Что ж,— заключил Мегрэ,— будем считать, что компьютер внес свой вклад в решение проблемы общения. И этот вклад проявился наиболее рельефно именно в вычислительных сетях.

Внимательный читатель наверняка заметил, что при ошибке в служебном кадре, который несет квитанцию, передача остановится — передающий узел так и не узнает судьбу переданного кадра и будет безнадежно ждать «у моря погоды». На этот случай предусмотрен механизм так называемого *тайм-аута*. Он состоит в том, что передающий узел ждет квитанцию строго определенное время (это и есть время тайм-аута), после чего он снова отправляет кадр, на который не получено квитанции. Легко заметить, что при утере положительной квитанции в сети появится второй дубликат кадра, так как первый уже ушел к адресату. Таких дубликатов может появиться много, если нарушен обратный канал и он не пропускает кадры с квитанциями. Поэтому в вычислительной сети предусмотрена еще служба «отлавливания» и уничтожения всех дубликатов кадра и аварийной сигнализации при появлении подряд нескольких тайм-аутов, что свидетельствует о неисправности канала связи.

И еще один важный механизм управления потоком кадров в канале связи между двумя узлами. Это механизм окна. *Окном* называют число кадров, переданных без подтверждения. До сих пор была описана система связи с единичным окном. Но при большой задержке в канале (что бывает при передаче на большие расстояния) передающему узлу приходится долго простаивать в ожидании квитанции. Так, например, бывает при использовании спутникового канала связи, где задержка равна 0,2 с. Если действовать по-старому, то по такому каналу можно передавать со скоростью один кадр в 0,4 с (0,2 с на передачу кадра и столько же на получение квитанции). Именно в этом случае следует иметь большое окно, т. е. передавать подряд кадры без подтверждения, но сохранять их все в памяти. Число таких кадров в спутниковом канале может доходить до 100 и более. При получении положительной квитанции следует продолжать передачу, вводя в память новые кадры на место поврежденных. При отрицательной квитанции можно или повторить передачу начиная с ошибочного кадра, что внесет задержку в несколько кадров (окно), но не нарушит порядка их передачи, или повторить только ошибочный кадр, что приведет к нарушению порядка их передачи при задержке лишь одного кадра.

СНОВА ПРОТОКОЛЫ

Далее защищенный кадр обрамляется *флагами,* которые имеют вид 01111110— шесть единиц между двумя нулями. Эти флаги нужны для того, чтобы определить, где начало и конец кадр-ра, так как в канале кадры идут без интервалов. А чтобы комбинация флага не встретилась в теле кадра, после каждой пятой единицы подряд вставляется 0. Эту операцию называют *битстаффингом* — вставкой бит. На приемном конце в процессе приема идет счет единиц, следующих подряд. Если после пяти единиц стоит 0, то он сбрасывается (это результат битстаффинга), а если 1, то принимается решение, что это флаг, разделяющий кадры.

И, наконец, на 1-м (физическом) уровне происходит передача кадра в физический канал связи (телефонный, оптический, радиорелейный, тропосферный, спутниковый и т. д.) через аппаратуру передачи данных (модем, усилитель и т. д.). Например, при передаче по телефонным каналам модем кодирует 0 одной звуковой частотой, 1— другой, а сигнал наличия связи — третьей. На приемном конце телефонного канала такой же модем работает в режиме демодуляции и восстанавливает переданные нули и единицы кадра.

Все описанные процедуры и есть пример реализации основных протокольных правил, которые необходимо выполнять при общении пользователей и компьютеров через вычислительную сеть.

Рассмотренная семиуровневая система протоколов нигде обычно не применяется во всей полноте. Да это и не нужно: каждая конкретная сеть имеет свою усеченную систему тех протоколов, которые нужны именно для этой сети.

ТИПЫ ВЫЧИСЛИТЕЛЬНЫХ СЕТЕЙ

Их много, но основными по конфигурации являются узловые, моноканальные и кольцевые (или циклические) сети.

Узловая сеть состоит из узлов коммутации и каналов связи (рис. 17, *а*). В узлах стоят быстродействующие компьютеры (их называют коммутационными машинами), которые проверяют правильность передачи кадра с помощью информации, заложенной в его концевик (в соответствии с протоколом 2-го уровня), определяют по заголовку пакета и имеющейся в узле информации о состоянии сети дальнейший маршрут (протокол 3-го уровня) и отправляют кадр дальше, сопроводив его новой служебной информацией для следующего узла в соответствии с протоколом 2-го уровня. Так от узла к узлу передаются кадры (их чаще называют пакетами), пока они не достигнут узла, к которому подключен адресат — компьютер, которому адресовано сообщение. Происходит это очень быстро и надежно, ошибочная передача

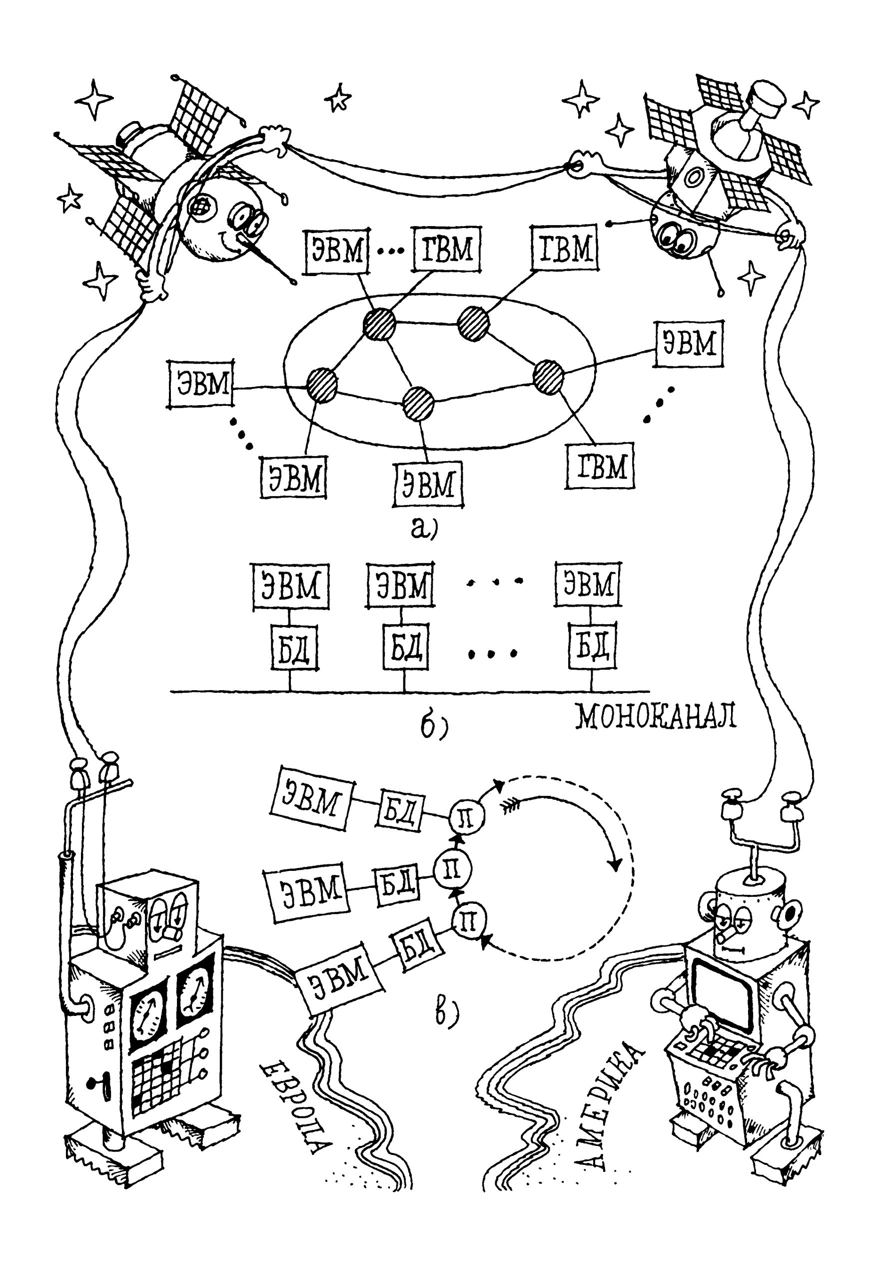

ЭВМ
ГВМ
ГВМ
ЭВМ
ЭВМ
ЭВМ
ЭВМ
ГВМ
а)
ЭВМ
ЭВМ
ЭВМ
БД
БД
БД
МОНОКАНАЛ
б)
ЭВМ
БД
П
ЭВМ
БД
П
ЭВМ
БД
П
в)
ЕВРОПА
АМЕРИКА

Рис. 17

между узлами исключается. А при выходе из строя одного узла сети кадры направляются в его обход.

Примером такого рода сети является сеть ARPA (США) — первая вычислительная сеть. Она раскинулась на всей территории США и имеет более 60 узлов. В эту сеть входят узлы на Гавайских островах и в Европе, связанные с Америкой через спутниковые каналы. Среднее время передачи пакета в сети между двумя абонентами 0,2 с, а между самыми удаленными 0,5 с. Вычислительная мощность сети образуется 150 ГВМ, а ее центр управления расположен в Кембридже (шт. Массачусетс).

У нас в Союзе уже давно эксплуатируется специализированная узловая вычислительная сеть для продажи авиабилетов «Сирена». Кто пользовался ее услугами, тот знает, как быстро она позволяет обращаться к банкам данных на ЭВМ, расположенным в других городах, где есть узлы «Сирены». В банках данных «Сирены» хранятся сведения о номерах свободных мест на все самолеты, вылетающие из этого города, на 30 дней вперед. Обращение к такому банку позволяет получать информацию о наличии свободных мест на нужный рейс, забирать и бронировать их.

Моноканальные сети используют лишь один канал связи (рис. 17, *б*), объединяющий всех абонентов сети. Это может быть коаксиальный кабель, тропосферный канал (обычная радиосвязь), спутниковый канал и т. д. Каждый абонент может послать свой пакет в моноканал. Этот пакет принимают все абоненты, убеждаются в его правильности и по адресу определяют, кому он предназначен. Адресат продолжает обрабатывать этот пакет в соответствии с протоколами более высокого уровня, а остальные просто игнорируют его. Основной трудностью при работе вычислительной сети с моноканалом является возможность наложения пакетов, т. е. одновременный выход в сеть двух и более абонентов со своими пакетами. Это всегда возможно, если между абонентами нет предварительной договоренности о времени выхода на моноканал. Указанная трудность преодолевается введением блоков доступа (БД), через который каждый абонент выходит в моноканал. Блок доступа, получив заказ на передачу пакета, «прослушивает» канал и при отсутствии чьей-либо передачи отправляет свой пакет в моноканал, продолжая его прослушивать. Если одновременно начали передачу два и более абонентов, то они, обнаружив это, прекращают передачу и каждый возобновляет ее лишь через случайный интервал времени (и при отсутствии чужой передачи, разумеется). Здесь случайность вводится намеренно, чтобы не допустить синхронизации момента выхода абонентов на моноканал. Такой способ получил название *случайного доступа* и широко используется ввиду его простоты и надежности.

Однако при случайном доступе неизбежно будут и простои канала связи — они составляют примерно 60 % всего времени работы. При регулярном доступе этого можно избежать, если,

например, предварительно опрашивать всех абонентов об их потребностях в передаче и указывать им момент их выхода на моноканал. Но это сложнее и требует организации специальной диспетчерской службы на одной из ЭВМ, входящих в сеть.

И наконец, *кольцевые* (циклические) *сети*. Они являются в определенном смысле гибридом узловых и моноканальных сетей. Канал связи такой сети представляет собой замкнутое кольцо (рис. 17, *в*) из повторителей (П), связанных коаксиальными или оптическими каналами. Роль повторителя сводится к задержке поступившего пакета на время, необходимое для прочтения его адреса. Если адрес чужой, то пакет свободно уходит по каналу связи к следующему абоненту. А если пакет адресован этому абоненту, то он не повторяется и тем самым изымается адресатом из сети. Легко заметить, что повторитель играет роль простейшего узла коммутации, который или пропускает, или не пропускает пакет. Отправка новых пакетов каждым абонентом происходит через блок доступа в момент «молчания» канала. Доступ к каналу связи при этом облегчается. Приходящий пакет, который может «накрыть» тот, который передается, можно легко «придержать» в памяти повторителя до окончания передачи.

Описанные типы сетей возникли и используются для удовлетворения определенных государственных потребностей промышленности, обороны, управления и т. д. Но некоторые из них создаются для удовлетворения сугубо личных информационных потребностей, возникающих у самого широкого круга людей, совсем чуждых вычислительной технике. Эти потребности связаны, например, с необходимостью получить справку о том, что и когда идет в местном кинотеатре, как сыграла любимая футбольная команда в чемпионате страны, где купить лыжи и сколько они стоят и т. д. и т. п. Возможность быстро получить такого рода информацию и обеспечивают специальные сети. Одну из них называют...

Телетекст

Вычислительная сеть — очень сложное образование компьютерной техники. И основной трудностью здесь является организация двусторонней системы связи между компьютерами. Для этого прежде всего нужно иметь канал двусторонней связи, например телефонный.

А нельзя ли для этого воспользоваться еще более простой односторонней связью? Например, существующей системой телевидения, более массовой, чем телефонная сеть. Оказывается, можно. При этом задачи, решаемые с помощью такой вычислительной сети, ограничены, но тем не менее очень полезны в повседневной жизни. Речь идет о передаче по стандартному телевизионному (ТВ) каналу на стандартный телевизор информации в виде текстов, интересующих пользователя. Ввиду отсутствия обратной связи ТВ-станция должна передавать все, что может заинтересовать различных пользователей.

Как видно, это вычислительная сеть с моноканалом, где каждый ее абонент только принимает информацию. Так как нет канала обратной связи, то передача

должна содержать все, что нужно или может понадобиться ее абонентам А так как их много и потребности их разнообразны, то следует передавать все время и с большим запасом, чтобы в любой момент абонент мог получить нужные ему сведения.

Что это за информация? Она ориентирована на широкого пользователя: местные и международные новости, расписание передач радио и телевидения, расписание движения транспорта (автобусов, поездов, самолетов), что, где и когда идет в кинотеатрах, концертных залах и театрах, реклама и многое другое, что может понадобиться абоненту в его беспокойной жизни.

Эта информация поступает по ТВ-каналу связи непрерывно, точнее после передачи каждого кадра, когда проходят так называемые неактивные строки, которые гасятся и не попадают на экран — их четыре на каждый кадр. Именно эти строки используются для передачи текста в системе Телетекст. Каждая ТВ-строка в системе содержит 40 символов (букв, цифр, значков и т. д.); 24 строки образуют одну страницу телетекста, которая полностью умещается на экране. Полная передача включает до 800 стр., которые передаются в течение 3 мин 20 с и непрерывно повторяются (это данные о Британской системе телетекста Prestel, которая вступила в строй в 1979 г.).

Чтобы стать абонентом этой системы, достаточно приобрести небольшую приставку к своему телевизору. Она представляет собой несложное электронное устройство для «выуживания» нужных текстовых строк из ТВ-передачи по заказу абонента, что осуществляется с помощью простой клавиатуры, расположенной на этой приставке. В результате нужные страницы передаваемого телетекста при появлении лягут в память приставки и смогут быть выведены на ТВ-экран по желанию абонента. На всю процедуру приема и заполнения памяти уйдет не больше одного цикла передачи (в системе Prestel не более 3,5 мин). Полученная информация хранится приставкой сколь угодно долго и может быть в любой момент визуализирована на ТВ-экране или заменена на новую по другому запросу абонента.

Отсутствие обратной связи в системе Телетекст исключает применение переспроса для устранения ошибок (об этом мы говорили в разделе о протоколах). Как же здесь устранить ошибки, вызванные помехами, которые неизбежны при передаче по ТВ-каналу — их часто прямо видно на экране?

Борьба с помехами ведется и здесь. Делается это так. Каждый символ передается одним байтом, т. е. восемью битами Из них семь информационные, что позволяет закодировать $2^7=128$ знаков. Этого вполне достаточно для передачи текстовой алфавитно-цифровой информации. А восьмой бит контрольный — он проверяет семь предыдущих на четность. Если их сумма четна, то на восьмом месте ставится нуль, например 01101010, а при четной сумме — единица, например 10110111. При приеме несложная схема проверяет соответствие первых семи символов восьмому. И при несоответствии делает пропуск в тексте. Если это была буква, то она, как правило, легко восстанавливается по смыслу, так как буквенный текст всегда избыточен. Например, получив на экране РЕП РТУАР, нетрудно понять, что пропущена буква Е. Если же это сделать не удается (например, при приеме цифровой информации), то следует просто повторить прием нужной страницы.

Описанная система применима лишь для абонентов, чьи информационные

потребности легко прогнозируются. Но при необходимости получить что-то выходящее за рамки передач телетекста следует воспользоваться другой массовой системой, которую называют...

Видеотекс (без т)

Эта система имеет обратную связь, для чего используется обычный телефон, позволяющий осуществить двусторонний обмен данными. Абонент здесь должен иметь уже три элемента: телевизор, телефон и приставку видеотекса. Телевизор используется только как монитор, с помощью которого отображается на экран принятая по телефону информация. Приставка позволяет осуществить заказ нужной страницы текста и получить эту страницу с помощью телефона.

Делается это так. Вы соединяетесь по телефону с компьютером, в памяти которого имеется интересующая вас информация, и кладете трубку на приставку видеотекса (для этого на ней есть удобное углубление). Набираете номер нужной вам страницы, и ваш запрос поступает в компьютер, который находит эту страницу и тут же отправляет ее в память вашей приставки. Так как связь по телефону медленная (75 бит/с), то вы получите свою страницу лишь через 1,5 мин. Но это все же вдвое быстрее, чем при работе с системой Телетекст. Номер же нужной вам страницы легко узнать, запросив предварительно каталог того, что есть в памяти этого компьютера.

Обе системы (телетекст и видеотекс) легко совместимы в одной приставке и хорошо дополняют друг друга, как оперативная и внешняя памяти компьютера. Здесь вся изменяющаяся ежедневно и ежечасно оперативная информация идет по телетексту, т. е. по ТВ-каналу связи, и обращение к большим массивам информации происходит через видеотекс, т. е. по телефонному каналу связи. Объем же получаемой при этом информации одинаков — одна или несколько страниц текста (такова, например, Британская служба Prestel).

ИЕРАРХИЯ СЕТЕЙ

Вычислительные сети, как всякая сложная система, имеют иерархический характер, т. е. устроены так, что малые сети входят составной частью в большие вычислительные сети. Иерархия сетей такова.

Глобальная вычислительная *сеть* объединяет абонентов, расположенных в различных странах или даже на разных континентах. Строится глобальная сеть обычно на спутниковых каналах связи, позволяющих связывать узлы сети и ее абонентов, расположенных на расстояниях до 10—15 тыс. км.

К глобальным сетям подключаются *региональные сети,* которые связывают абонентов, расположенных на расстоянии 10—1000 км, т. е. работающих на территории города, района, области или даже небольшой страны.

Глобальные и региональные сети имеют узловую структуру с мощными узлами и каналами связи с высокой пропускной способностью.

К региональным сетям подключаются *локальные сети,* або-

ненты которых находятся на небольшом расстоянии друг от друга. Обычно локальные сети связывают пользователей, расположенных в одном или нескольких близко расположенных зданиях. Однако при использовании радиоканалов связи локальные сети могут связывать абонентов, удаленных на расстояние до 20 км (например, таксистов). Локальные вычислительные сети получили в последнее время бурное развитие. Они широко используются на предприятиях, в учреждениях, конторах. Объясняется это тем, что 80—90 % всей появляющейся информации (в науке, управлении, на производстве) циркулирует и обрабатывается поблизости от мест ее появления — в институтах, учреждениях, на предприятиях. И только 10—20 % этой информации связано с внешними взаимодействиями этих организаций. Локальные сети могут иметь любую структуру, но чаще всего используется моноканальная и кольцевая. В отличие от региональных и глобальных сетей локальные вычислительные сети могут иметь в качестве абонентов не только ЭВМ, но и их отдельные компоненты: процессоры, терминалы, устройства внешней памяти и т. д.

Примером наиболее распространенной локальной вычислительной сети является Ethernet (США) с моноканалом на коаксиальном кабеле, который обеспечивает передачу информации со скоростью 10 Мбит/с (1 Мбит $=10^6$ бит). Длина сегмента кабеля без промежуточных усилителей 500 м, а общая длина — до 2500 м. Сеть имеет 1024 станции, через которые к кабелю подключаются ее компоненты, и 100 приемопередатчиков. Максимальная длина пакета в сети 1522 байт.

И, наконец, очень перспективными являются *интегральные* вычислительные *сети,* позволяющие абонентам не толко обмениваться вычислительными ресурсами, но и передавать по сети речь и изображения. Трудность здесь заключается в том, чтобы для передачи речи и изображений использовать те же информационные пакеты, которые применяются для передачи сообщений. С решением этой задачи (а она будет решена, так как и речь и изображение всегда можно представить в виде сообщений) отпадает необходимость в самостоятельных сетях для телефонной, телеграфной, а может быть, и телевизионной связи. Их функции возьмут на себя интегральные вычислительные сети. Заметим, что любая вычислительная сеть уже может выполнять функции телеграфа: текст телеграммы ничем не отличается от сообщения, которое передается по вычислительной сети с целью ее выдачи на дисплей или принтер адресата. Эту службу вычислительной сети называют электронной почтой.

Вычислительные сети — наиболее сложные и наиболее удобные образования века вычислительной техники. Этот сервис получен дорогой ценой сложности. Но других путей пока нет,

— Я хорошо пониманию,— сказал Мегрэ,— что вычислительные сети — сверхсложные системы. Если суперкомпьютер — очень сложная машина, то их объединение в сеть неизбежно создает сверхсложность. И именно это мне кажется угрожающим. Действительно, грандиозность этих систем должна неизбежно приводить к их ненадежности. У меня нет уверенности, что такого рода система реагирует всегда адекватно.

— Вы и правы, и не правы, шеф. Ведь ненадежность бывает разная. Одна связана со случайными сбоями и отказами, неизбежными во всякой действующей системе. Чем больше и сложнее системы, тем значительнее влияние таких случайных помех. Это самый простой вид ненадежности, и ее легко выявить по разной реакции системы на одинаковые сигналы в одинаковых условиях. Введение избыточности и переспроса решает проблему ненадежности такого рода.

— Вы что хотите сказать, что любую сколь угодно большую и ненадежную систему можно сделать надежной, только дублируя ее элементы и организуя переспрос в коммуникации?— изумился Мегрэ.

— Да, но устранить так можно только ненадежность, вызванную лишь случайными помехами,— ответил Поль.

— А разве есть еще какая-то ненадежность?

— Есть, и она, к сожалению, устраняется не столь просто. Это функциональная ненадежность. Грубо говоря, система на определенные воздействия реагирует одинаково, т. е. не случайно, но не так, как следует. Эта ненадежность свойственна только очень сложной системе.

— Но почему же такую неправильую реакцию не устранить в самом начале, на этапе отладки системы?— спросил Мегрэ.— Что может быть проще: проверить все реакции на все возможные воздействия и убедиться, что система работает правильно или исправить ее при неправильной реакции.

— Все это хорошо лишь для простой системы. А для сложной число всех реакций так велико, что не поддается проверке. Например, при 100 двоичных входах (они принимают лишь два значения 0 или 1) число возможных различных ситуаций на входе 2^{100}. Чтобы убедиться, что эта система (а это не очень-то и сложная система, процессор любого компьютера значительно сложнее) работает правильно, нужно проделать с ней 2^{100} экспериментов. Много это или мало? Если делать один эксперимент в секунду, то для того, чтобы выполнить все $2^{100}=10^{30}$ экспериментов, нужно потратить более 10^{12} лет — тысячу миллиардов лет! А наша Вселенная существует всего 20 млрд. лет.

— Не пугайте меня этими трудностями, Поль,— усмехнулся Мегрэ,— ведь всего-то проверять не нужно. Так что ваши миллиарды лет пойдут впустую.

— Вы не правы, шеф,— обиженно сказал Поль.— Все зависит от того, кто использует результат, полученный из-за ошибочной реакции системы. Если она иногда выдает вам неправильный результат, то все зависит от того, насколько он неожидан для вас. При грубой ошибке вы легко обнаружите это и не воспользуетесь полученным решением. Если же ошибка невелика, то она не принесет большого вреда. Вот и получается, что функциональная ненадежность не страшна, если ее последствия проходят через фильтр здравого смысла, т. е. через человека. Именно так, например, используется почти вся компьютерная техника.

— Ну и бог с ней, с этой ненадежностью. Стоит ли об этом много говорить!— махнул рукой Мегрэ.

— Действительно, о ней можно было бы не беспокоиться, если бы не очень распространенный случай — сложная система управляет реальным объектом, минуя человека. При этом ошибочное решение будет сразу реализовано в объекте и ... все зависит от этого объекта. Если управляемый объект может быть опасен, как, например, ядерный реактор, то управляющая система должна быть очень надежна.

— Ну, ну! Я уверен, что ядерным реактором управляет очень надежный компьютер, решения которого к тому же находятся под контролем людей.

— Да, здесь система управления достаточно проста, в конечном счете все сводится к решению: вводить или не вводить в реактор графитовые стержни, которые поглощают нейтроны и тем самым гасят ядерную реакцию. Но есть системы управления гораздо более сложные. Например, СОИ.

— Это, что, Стратегическая Оборонная Инициатива США?— оживился Мегрэ.— О ней сейчас много говорят как о щите, который будет надежно защищать США от стратегических ракет.

— Вот именно надежность ее и подвергается серьезному сомнению. Речь идет о функциональной надежности, разумеется. Дело в том, что СОИ будет управляться сложнейшей вычислительной сетью, разбросанной по всей территории США. Число возможных ситуаций, которые могут сложиться в этой системе, громадно и не поддается учету. А решения этой сложнейшей системы реализуются в виде взрывов ядерных бомб для «накачки» лазеров, лучи которых должны поражать летящие цели. Человека здесь нет и не может быть, решение должно быть принято в доли секунды, автоматически. Источником функциональной ненадежности такой системы управления является невозможность даже представить все возможные варианты ситуаций, которые сложатся в СОИ. А поэтому нет гарантии, что на какую-то «нештатную» ситуацию СОИ из оружия защиты не превратится в оружие нападения (случайного в данном случае). Именно так сложные системы автоматического управления становятся причиной тяжких катастроф.

— Выходит, что нужно избегать создания сложных систем автоматического управления реальными объектами,— удивился Мегрэ.— А как насчет столбовой дороги прогресса, который требует 100 %-ной автоматизации технологии, управления, решений...?

— Здесь,— улыбнулся Поль,— мы (человечество) подошли к некой границе, пересекать которую нельзя. Автоматизация решений — это, пожалуй, самая опасная область компьютерного мира. Не все решения нужно автоматизировать. Это может быть опасным для человечества.

II. ДИАЛОГ

8. ПОГОВОРИ-КА ТЫ СО МНОЙ...

(Диалог как общение)

Несомненно, диалог с компьютером должен быть подобен беседе двух людей. В такой беседе очень часто бывают важны даже самые тонкие нюансы — жест, взгляд, интонация, вздохи, подмигивания и т. д. Иногда они несут значительно больше информации, чем сказанные при этом слова, или сильно искажают их смысл. Однако не это интересует нас сейчас — вряд ли в ближайшее время у нас появится возможность в диалоге с компьютером использовать подобные средства общения (а жаль — это в самом деле позволило бы при общении с машиной достичь понимания с полуслова).

ДИАЛОГ. ЧТО ЭТО ТАКОЕ!

Здесь мы несколько отвлечемся от диалога с машиной и будем говорить в основном о диалоге между людьми. Заглянем немного и в психологию — именно эта наука больше всего занимается вопросами общения (коммуникации) между людьми. А диалог — одна из форм такого общения. Автор не психолог и понимает, насколько рискован такой экскурс в не свою отрасль знаний, к тому же в науку, связанную с работой мозга, где еще так много нераскрытых тайн. Однако очевидно, что нельзя создать сколь-нибудь удовлетворительную систему диалога человека и машины, не используя хотя бы некоторые представления о мышлении человека, работе мозга, которые накоплены в психологии.

Для начала заглянем в Большую советскую энциклопедию и познакомимся со статьей «Диалог». Там указано, что в переводе с греческого это слово означает «разговор, беседа» и употребляется для определения такого вида речи, для которого характерно:

ситуативность — зависимость от обстановки, в которой протекает диалог;

контекстуальность — обусловленность предыдущими высказываниями;

заранее не запланированный характер.

А теперь посмотрим, что под диалогом понимают представители различных наук.

МНЕНИЕ ПСИХОЛОГА

Психологи называют диалогом такое речевое общение двух партнеров (разумеется, людей), когда они попеременно говорят, слушают и, разумеется, осмысливают как услышанное, так и сказанное ими. Считается, что активность партнеров диалога приблизительно одинакова, и если один из них отказывается от своей очереди взять слово, то диалог превращается в монолог. Если отказываются оба, то разговор исчерпывается. Поведение каждого человека в диалоге удобно разбить на такты: один такт — речь, другой — молчание. Соотношение времен между тактами называют ритмом диалога.

Уже в 1939 г. американский ученый Э. Чаппл с помощью экспериментов установил, что ритм диалога для каждого человека — относительно устойчивая характеристика. Этот ритм индивидуален, а различие в ритмах между людьми весьма велико. Так были определены ритмы диалога у 154 продавцов большого универмага. С каждым из них по одной и той же методике около часа беседовал экспериментатор. По результатам эксперимента его участники были разбиты на три группы. Интересно, что в первую группу (с наибольшим преобладанием разговора над молчанием) вошли лучшие продавцы.

Сказанное не означает, что ритм диалога нельзя изменять. Обычно в диалоге в отличие от интервью партнеры стараются подстраиваться друг к другу, и, так как ритм диалога достаточно гибок, это им вполне удается. Однако бывает, что продолжать разговор можно только ценой «отказа» от своего ритма, в результате усилия над собой. Все такие «отказы» за какой-то период времени, например за сутки, настолько накапливаются, что начинают «взывать к компенсации». Поэтому, если человеку на работе необходимо много разговаривать, то после работы ему приятнее отмалчиваться (или участвовать в разговоре формально) и наоборот.

Однако не только одним ритмом характеризуется диалог. Американский психиатр Дж. Джаффе исследовал диалог с точки зрения смены состояний. Он выделил четыре состояния диалога:

1) оба партнера говорят;
2) оба партнера молчат;
3) первый партнер говорит, второй молчит;
4) второй партнер говорит, первый молчит.

Очевидно, что в каждый момент возможно только одно из этих состояний. В эксперименте перед каждым партнером диалога был микрофон, а специальное устройство, регистрируя звуковые колебания, фиксировало то или иное состояние. Результаты экспериментов были проанализированы с помощью ЭВМ, и когда определили, как часто (или с какой вероятностью) одни состояния следуют за другими, то оказалось, что смена состояний в диалоге описы-

вается цепью Маркова. Это значит, что вероятность появления каждого из четырех состояний зависит только от того, в каком из этих состояний находился диалог непосредственно перед данным моментом, и не зависит от всех предыдущих состояний. Эксперименты подтвердили, что подобная закономерность справедлива не только для взрослых, но и для детей. Можно считать, что это свойство диалога и сделало его наиболее устойчивой первичной формой общения между людьми.

Оказывается, что между диалогом и монологом существует четкая связь. Советский психолог Л. С. Выготский поставил простой и изящный эксперимент. Он поместил ребенка в общество детей, которые не могли его понять,— глухонемых или говорящих на другом языке. Казалось бы, ребенок начнет говорить только «для себя», т. е. в монологе. Но этого не произошло. В привычных ситуациях он вел диалог с детьми, которые его не понимали. Речь «для себя» возникала, лишь когда у ребенка возникали затруднения. Отсюда следует важный вывод: сначала ребенок овладевает диалогом, связывающим его с внешним миром, а уж только потом эта речь «расщепляется» на речь для себя и речь для других. По мере взросления речь для себя сменяется внутренней речью. У взрослых также внутреннее проговаривание часто сопровождает усиление мысли. Л. С. Выготский считал, что эгоцентрическая, направленная на себя, речь ребенка очень легко становится мышлением в собственном смысле этого слова: с ее помощью ребенок учится планировать свои действия и решать задачи. Позднее человек обретает навыки монолога. И лишь немногие взрослые владеют монологом столь совершенно, что могут без подготовки выступить на собрании, комментировать событие, «с ходу» написать заметку в стенгазету, составить документ и т. д. Диалогом же мы все владеем в совершенстве.

Первичность диалога по сравнению с монологом фактически вытекает из социального бытия человека, из необходимости его эффективного общения с другими людьми, без которого человек не только не может быть человеком, но и не может существовать вообще.

Известный рассказ Р. Киплинга о Маугли — лишь прелестная выдумка писателя, которую жизненный опыт опровергает. Дети, которые в начале жизни лишены возможности учиться диалогу (а таких случаев было много), никогда не становятся полноценными людьми. Может быть, поэтому Киплинг и закончил свой рассказ именно в тот момент, когда Маугли предстояло стать человеком — влиться в человеческое общество. Мудрый писатель наверняка чувствовал, что это невозможно.

Некоторые психологи считают самой характерной особенностью диалога возможность собеседников прерывать друг друга. Именно этим обусловлен быстрый темп речи: говорящий торопится досказать мысль, пока его не прервали. Поэтому он не

«шлифует» высказанные мысли, иначе возникнут паузы, которыми сможет воспользоваться партнер. А раз паузы сокращаются, некогда тщательно подбирать слова. Реплики короткие, незавершенные, но партнеры понимают друг друга потому, что они взаимообусловлены, связаны с контекстом, с ситуацией разговора. Установлена еще одна интересная зависимость диалога: чем длиннее реплика одного партнера, тем короче ответ, и наоборот.

Типичным монологом является доклад, все вопросы задаются после его окончания. К слушанию монолога можно отнести и чтение книги. Однако и здесь проявляется диалог: если доклад или книга вызывают интерес, у слушающего или читающего возникает желание включиться в диалог. Слушая доклад, он обычно стремится включиться в диалог, т. е. высказывает реплики (вслух или про себя) или набрасывает план для выступления по теме доклада, а при чтении книги ведет мысленный диалог или делает заметки на полях. Так что монолог чужд человеческой натуре, и при первой возможности человек переходит на диалог.

Возвращаясь к диалогу с ЭВМ, можно добавить, что процесс выполнения программы на компьютере можно рассматривать как монолог, а процесс программирования с активным взаимодействием пользователя и машины, в ходе которого пользователь исправляет указанные машиной ошибки и ... допускает новые, естественно назвать диалогом.

МНЕНИЕ ИНЖЕНЕРА

Инженер относится к диалогу иначе, чем психолог. Для него основной категорией является информация, процессы ее возникновения и переработки. Поэтому для инженера важно построить формальную модель диалога. Это нужно прежде всего для понимания явления, для его моделирования и для использования полученной модели при создании диалоговых систем самого разнообразного назначения: для управления, обработки информации, проектирования, моделирования и т. п.

С формальной точки зрения диалог — это прежде всего процесс обмена информацией между двумя системами (собеседниками) в режиме реального времени. Основным является то, что диалог ведется о предметах, относительно которых у участников есть некоторые модели — представления об этих предметах. Слово «информация» здесь употребляется в обычном, обыденном смысле — «новые сведения или знания об окружающем мире».

Следует подчеркнуть, что в диалоге обмен информацией осуществляется не вообще (как в светской болтовне), а относительно строго определенного предмета (объекта). При этом предполагается, что в ходе диалога оба собеседника (или хотя бы один из них) преследуют относительно предмета беседы

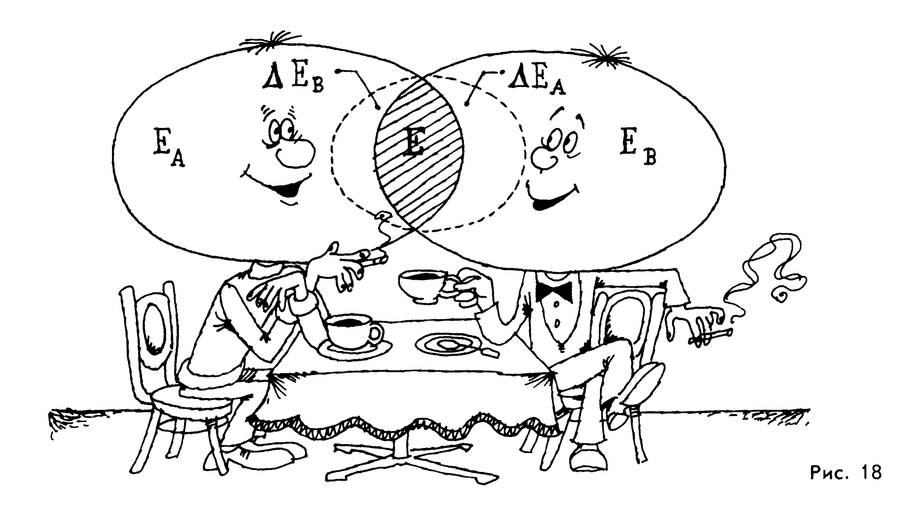


Рис. 18

определенные цели, которые их (его) волнуют, или во всяком случае собеседники не безразличны к предмету обсуждения. Такое представление о диалоге лучше позволит раскрыть очень условная схема (рис. 18). Знания двух собеседников (A и B) о предмете диалога здесь отображены в виде эллипсов E_A и E_B, причем под словом «знания» будем понимать сведения самого широкого круга: закономерности, факты, гипотезы, цели и т. д. (точнее, отражение всего этого в мозгу человека в виде модели-представления). Некоторые знания о предмете беседы как у одного (A), так и у другого (B) собеседника должны совпадать (эти общие знания представлены на рис. 18 в виде заштрихованной области E), иначе они не поймут друг друга.

Мы знаем, что разные лица, получив одно и то же сообщение, извлекают из него разную информацию. Например, известие о том, что найдены новые элементарные частицы — кварки, будет по-разному воспринято физиком и историком. Поэтому предположим, что у каждого собеседника могут быть свои интересы относительно информации, имеющейся у другого. Тогда каждый из собеседников извлекает из получаемых в процессе диалога сведений ту информацию, которая его интересует, расширяет свой запас знаний о предмете диалога (на рис. 18 штриховой линией).

Теперь посмотрим, какие же должны быть формальные условия для успешного диалога. Можно констатировать по крайней мере три условия, выполнение которых необходимо, чтобы диалог был результативным.

Условие первое, с которым читатель, наверное, сразу согласится: диалог имеет смысл вести тогда, когда знания обоих собеседников по обсуждаемому вопросу различны, не совпадают

полностью: $E_A \neq E_B$, иначе обмен информацией терял бы смысл. Но понятие смысла достаточно широко. Психологам известен такой тип общения, связанный с самовыражением, когда собеседников интересует не чужая новая информация, а лишь передача своей, независимо от того, нова она для собеседника или нет. Эта модель разговора, в ходе которого почти не слушают, а лишь говорят. Трудно возражать против такого «диалога», который по сути своей близок к монологу. Он иногда играет важную роль для поддержания психологически нормального климата, когда одному или обоим собеседникам хотя бы ради самоутверждения нужно выговориться. Но к рассматриваемому нами диалогу такой «диалог» никакого отношения не имеет — обмена информацией здесь не происходит.

Условие второе также интуитивно ясно: предметом диалога могут быть лишь такие объекты, о которых каждый собеседник хоть что-нибудь знает, т. е. $E_A \neq 0$ и $E_B \neq 0$. Причем обязательно должно существовать общее знание E (см. рис. 18). В противном случае собеседники попросту могут не узнать, что говорили об одном и том же предмете. Это условие может быть записано в следующем виде: $E_A \cap E_B \neq \emptyset$. Эта формула означает, что пересечение двух множеств (знаний) E_A и E_B о предмете диалога не должно быть пустым. Символ $\emptyset$ обозначает пустое множество, которое не содержит ни одного элемента.

Любопытно, что язык Линкос, разработанный для связи с внеземными цивилизациями по радиоканалу, опирается на то общее знание, которое есть у всех цивилизаций,— это устройство космоса, который мы наблюдаем.

В качестве третьего условия к диалогу естественно выдвинуть требование, чтобы объем знаний, который в процессе диалога получают партнеры, был бы максимальным. Только такой диалог может быть живым и активным. Это естественно, так как конечной целью диалога является получение информации. Поэтому эффективность диалога следует оценивать по приросту знаний, полученных собеседниками в процессе диалога. Таким образом, в процессе диалога должно происходить максимальное расширение объема знаний по обсуждаемому вопросу хотя бы у одного из партнеров диалога, т. е. должна решаться задача максимизации: $\Delta E \to \max$, где $\Delta E = E_1 - E_0$; E_0 — объем исходного знания по обсуждаемому вопросу; E_1 — объем знания после диалога.

Легко заметить, что для выполнения этого требования в процессе обучения часто пользуются монологом, т. е. лекцией, когда преподаватель «обрушивает» непрерывный поток информации на головы своих учеников. Но обильность такого потока вовсе не означает его интенсивного усвоения всеми слушателями. Именно поэтому такая устаревшая лекционная форма преподавания сейчас заменяется индивидуальной, где решающим фактором является диалог с преподавателем или компьютером, в процессе

которого ученик может задавать вопросы. Именно при этом будет максимизирован прирост знаний ученика.

СПОР КАК ДИАЛОГ

Еще древние считали, что «в спорах рождается истина». Спор — одна из форм диалога, и поэтому будет интересно и уместно проанализировать этот тезис, тем более, что он не бесспорен. В самом деле, кто задумывался о споре и участвовал в нем, тот наверняка заметил, что истина в спорах рождается крайне редко. Значительно больше при этом тратится времени. Если говорить точнее, то следует сказать так: «В спорах, вообще говоря, может иногда родиться какая-то истина» (здесь трудно не вспомнить очень язвительное определение спора в «Словаре сатаны» Алоиза: «Спор — это способ утвердить своего оппонента в его заблуждениях»).

Зачем тогда нужды споры? Неужели это лишь атавизм древних боевых поединков и турниров, принявших интеллектуальную форму? Нет! Споры нужны и даже необходимы. Но не для поиска истины — при этом за крохи истины приходится платить слишком дорого и самой ценной монетой — временем и хорошими взаимоотношениями. Споры нужны для другого.

В процессе спора рождается взаимопонимание и определяются точки зрения. Спорщики обычно спорят до тех пор, пока не поймут точку зрения своего противника, изменить которую, как правило, не удается. А если удается, то немедленно провозглашается рождение «истины». В действительности же это лишь другая точка зрения, к которой присоединился «обращенный».

Нас в споре будет интересовать не процесс «обращения», а процесс рождения взаимопонимания. Начнем с вопроса: а что значит «понимать друг друга»?

Простейшая модель понимания сводится к тому, что собеседники, употребляя в споре какие-то слова, используют их в одном и том же смысле. Это значит, что для обоих собеседников эти слова выражают одинаковые понятия. Если это не так, то взаимопонимания не будет. Например, слово «лук» определяет два понятия — растение и орудие стрельбы. И если собеседники, пользуясь этим словом, имеют в виду его различные понятия, то взаимопонимания не может быть в принципе. Вспомним скороговорку «Запер за́мок на замо́к, чтобы за́мок не замо́к», где двусмысленность, точнее трехсмысленность, слова «замок» и создает забавную ситуацию.

Каждый из нас прошел свой, индивидуальный путь развития, имеет свои собственные ассоциации и представления о мире. Именно поэтому мы используем свои собственные связи и отношения между словами и понятиями. Грубо говоря, смысл многих употребляемых слов у нас разный и очень индивидуальный, так как известно, что смысл слова задается только его употреблением.

А оно для каждого индивидуально. Эта индивидуальность может быть настолько значительной, что воспринимаемый смысл текста полностью искажается. Например, известно, что 10 % слушателей понимают своего собеседника «наоборот», т. е. вкладывают совершенно противоположный смысл в его сообщение. Стоит ли еще говорить об оттенках смысла, от восприятия которых, как известно, сильно зависит взаимопонимание?

Однако нельзя сказать, что мы плохо понимаем друг друга. Как это объяснить? Дело в том, что одна из важнейших функций спора состоит в том, чтобы выяснить, в каком же смысле применяет ваш оппонент слова, используемые в разговоре. Процесс такого выяснения и происходит в споре, что образует процедуру взаимопонимания. После этого вы можете изъясняться с собеседником на его «языке», понимать его и быть уверенным, что он понимает вас правильно.

Опыт показывает, что большинство споров возникает из-за непрерывного непонимания друг друга и немедленно прекращается по достижении взаимопонимания. Действительно, о чем спорить, если весь сыр-бор разгорелся относительно разных сторон одного и того же обсуждаемого предмета. Типичный пример: хорош или плох данный субъект? Одна сторона может приводить сколько угодно его хороших поступков, а другая — плохих, но спор от этого не разрешится — объект спора рассматривается с разных позиций. Совсем другой результат получится при обсуждении причин, почему этот субъект был плох в одной ситуации и хорош в другой.

Нужно ли говорить, как важно взаимопонимание в диалоге с машиной. (Заметим, что на неправильном понимании человека машиной построено много захватывающих сюжетов научной фантастики и ... анекдотов.) Что же в действительности? Здесь не может возникнуть спор, так как машине пока нечего делить с нами — у нее еще нет собственных целей (не исключено, что вскоре появятся, и тогда вопрос усложнится). Однако понимать человека машина должна в любом случае. Именно поэтому при проектировании диалога человека с компьютером необходимо предусмотреть «обсуждение» различных тем, в процессе которого налаживалось бы необходимое взаимопонимание.

Происходит это по-разному. Получив какую-то информацию, например задание от человека, машина проверяет, правильно ли она его «поняла». Для этого ей достаточно задать человеку один или несколько вопросов, в которых используется полученная информация. Например, получив распоряжение построить график, машина может понять это по-разному. Свое «понимание» она выразит вопросом «какого типа график?» или «где график?». В первом случае смысл распоряжения понят скорее всего правильно, а во втором — ошибочно.

Таким образом, только в диалоге может быть выяснен смысл

используемых слов, а следовательно, достигнуто понимание, что является основной проблемой при взаимодействии человека и компьютера. Диалог является самым эффективным средством налаживания взаимпонимания собеседников в любой области человеческой деятельности — политике (переговоры), экономике (рыночные взаимодействия), искусстве (выставки, концерты, читательские конференции и т. д.), спорте (соревнования) и т. д. Но это все взаимодействие людей, для которых диалог с подобными себе — привычное дело. Диалог же с компьютером — это новый вид налаживания взаимопонимания столь различных собеседников, какими являются человек и вычислительная машина.

— Из всех возможных диалогов,— заметил Мегрэ,— мне профессионально интересен лишь один — допрос. Диалог же с компьютером — это что-то уж очень новое, и пока мне не ясно, зачем мне вступать с ним в диалог. Ведь не допрашивать же мне компьютер!

— А почему бы и нет!— улыбнулся Поль.— Что такое допрос?

— Это,— менторски начал Мегрэ,— получение нужной и достоверной информации у человека. Допрашиваемый может не располагать этой информацией или не хотеть ею поделиться. И отличить эти два случая очень трудно. В этом и трудность допроса. Все искусство допроса и заключается в том, чтобы разобраться, какой случай имеет место, и в первом случае отстать от бедняги, а во втором убедить его не упираться и сообщить нужное (или «расколоть», как говорят на преступном жаргоне). Не так ли, дружище? Или вас нынче учат по-другому?

— Но, шеф, все, о чем вы сказали, относится и к диалогу с компьютером. Действительно, желая получить какую-то информацию, вы прежде всего должны быть уверены, что эта информация имеется в памяти компьютера. Как обрести эту уверенность? Да просто пытаться найти то, что нужно. Именно для этого нужен диалог, в процессе которого область поиска сужается до той, где ожидается интересующая вас информация. Не так ли и вы, комиссар, ведете допрос?

— Ну как вам сказать...— задумчиво сказал Мегрэ.— Я никогда не анализировал, как я допрашиваю. Просто ставлю вопрос так, чтобы ответ на него максимально приближал меня к цели, исключал бы все, что не имеет отношения к делу.

— А это и есть стратегия сужения области поиска.

— Вот уж не знал, что действую столь хитрым образом,— улыбнулся Мегрэ.— Но между человеком и компьютером большая разница. Человек может отказаться от ответа, и это его право, защищенное законом. Он всегда пытается предвидеть последствия, к которым приведет его ответ. Другое дело, что это предвидение может быть не самым точным. Но это его предвидение. В соответствии с этим он и отвечает, часто нарушая клятву говорить только правду. Именно для этого и предусмотрено в законе право не отвечать, чтобы не говорить неправду. А может компьютер солгать или отказаться от ответа?

— Да как вам сказать,— задумался Поль,— ложь как таковая не свойственна компьютеру. Ведь у него нет собственных целей, во имя которых ему пришлось бы лгать. Но эта ложь может быть заложена в его программу программистом. Вот типичная ситуация. Вы обращаетесь к компьютеру коллектив-

ного пользования за информацией, которая, как вам кажется, имеется в его памяти. Эта информация действительно там есть, но она засекречена ее хозяином, точнее, находится в его личном файле, доступ к которому происходит по «ключу»— кодовому слову, известному только хозяину файла. В ответ на ваш запрос компьютер отвечает, что такой информации у него нет. Солгал ли он? Вам кажется, что да, а по сути — нет. Действительно, содержимое засекреченного файла может быть доступно только при наличии «ключа». И если этот ключ не введен в компьютер, содержимое файла неизвестно, точнее, непонятно компьютеру. Так что он не лжет, когда отвечает, что не располагает нужной вам информацией, хотя для другого она находится. И никаким допросом «с пристрастием» вы не вытрясете из него эту информацию.

— Ну, это понятно,— кивнул Мегрэ,— ведь он только автомат и относиться к нему нужно, как к автомату.

— Компьютер, действительно, автомат,— улыбнулся Поль,— но общение с ним иногда очень напоминает общение с человеком. Вам, например, хорошо известно, что для того, чтобы «расколоть» иного беднягу, нужно подобрать к нему «ключик», т. е. такие аргументы, которые убедили бы его. При диалоге с компьютером такая ситуация складывается довольно часто. Вы должны знать «этикет» общения с компьютером и не нарушать его, в противном случае общение прекратится — компьютер просто не поймет вас.

— «Не понимать» или «не хотеть понимать»— это разные вещи,— заметил Мегрэ.— Ваш пример неудачен. Это равносильно тому, что я буду вести допрос на языке, не понятном подследственному. Естественно, что никакого разговора не получится. Меня здесь волнует другое. Может ли компьютер действовать на основе соображений, не понятных мне. Именно в этом случае я буду понимать его плохо.

— Вы можете быть спокойны, шеф,— улыбнулся Поль,— компьютер не является личностью (пока!). Его цели заложены в него программистом и направлены на эффективное взаимодействие человека с компьютером. Так что можете считать его вполне дружественным собеседником, который не имеет «задних мыслей» и «камней за пазухой», о которых любят говорить фантасты, описывая компьютеры будущего. Наш сегодняшний компьютер — симпатичный простак, который всегда рад услужить вам, если ему растолковать все достаточно подробно и понятно.

9. ПОЙМИ ХОТЬ САМОЕ ПРОСТОЕ!

(Проблема понимания естественного языка)

УРОВНИ ПОНИМАНИЯ

Рассматривая диалог «человек-компьютер», мы не должны забывать, что фактически речь идет о системе общения или, как сейчас принято говорить, о системе коммуникации. Поэтому необходимо ясно представлять себе, что общение может происходить на различных уровнях понимания между собеседниками.

Попробуем дать классификацию уровней понимания, чтобы

иметь возможность каким-то образом оценить достигнутый уровень общения и понимания (в том числе и при конструировании и исследовании диалоговых систем «человек — компьютер»).

Слово «понимание» мы применяем к самым различным ситуациям, в которых это слово имеет совершенно различные значения. Едва ли стоит говорить о том, что возможны различные степени понимания одного и того же сообщения. Достаточно напомнить хотя бы как зависит понимание такого сообщения, как, например, книга или кинофильм, от возраста «приемника». Как известно, «смотреть» не значит «видеть», и это различие зависит от очень многих факторов.

Прежде всего ограничим значение слова «понимание». Будем применять его только к таким ситуациям, когда участники диалога (их называют иногда коммуникантами) обмениваются определенно выраженными сообщениями в виде более или менее общепринятых знаков (слов, жестов, рисунков и т. д.), которые образуют «текст» сообщения. Тем самым исключаются такие ситуации, как «телепатическое общение» — прямой и непосредственный обмен мыслями между сознаниями коммуникантов, без какого-либо текста-посредника, а также общение путем изменения выражения лица или глаз человека. В этом случае мы не можем определенно указать наличие текста, к которому относится наше понимание, так как здесь трудно определить наличие каких-то знаков.

Несмотря на такое ограничение ситуаций общения, в значении слова «понимание» остаются еще широкие градации.

Первый уровень понимания

Для этого уровня понимания — назовем его *синтаксическим* — характерно, что «приемник» (тот, кто должен понимать) способен воспринять лишь структуру текста, т. е. вынести суждение о правильности его построения. Понимание здесь совсем не относится к содержанию этого текста. Понимающий имеет некоторую исходную информацию о правилах грамматики текстов, и не более. Связь текста с реальностью его совсем не «интересует». Знания «приемника» на этом уровне ограничиваются некоторой начальной информацией, позволяющей ему отличить только грамматически правильное предложение от неправильного. Единственным критерием истинности для такого коммуниканта будет грамматика (точнее, синтаксис), а не явления окружающего мира. Например, тексты «Волк ест козу» и «Коза ест волка» для него одинаково правильны. Этот уровень понимания вполне достаточен для проверки правильности способа выражения. Именно на этом уровне происходит понимание знаменитой фразы академика Щербы «Глокая куздра штеко бурдланула бокра и

курдречит бокренка», которая иллюстрирует важность знания синтаксиса для понимания текста.

Общение между любыми собеседниками на синтаксическом уровне означает, что они умеют соблюдать некий речевой ритуал: произносить приветствия, задавать вопросы типа «как поживаете?» и т. д. Кстати, если кто-то в ответ на последний вопрос всерьез станет рассказывать о своих делах, это вызовет только недоумение. В английском языке этот вопрос (How do you do?) давно уже стал стандартным приветствием, на которое принято отвечать тем же приветствием. Говоря по-русски, мы еще не можем на вопрос «Как поживаете?» отвечать теми же словами, но, быть может, к этому все идет.

Второй уровень понимания

Для этого уровня понимания — назовем его *первично-семантическим* — характерно, что воспринимающий способен не только расчленить данный текст на грамматические элементы и проверить их соответствие правилам грамматики, но и сопоставить им четкие элементы реальности.

На этом уровне «приемник» воспринимает жестко закрепленный за текстом смысл, но без учета окружающего контекста. Такое буквальное понимание свойственно ЭВМ, когда она получает текст программы на каком-либо алгоритмическом языке, например Алголе или Фортране, и сопоставляет каждому законченному фрагменту этого текста (знаку) некоторую машинную подпрограмму (модуль). Например, выражение A*B означает, что следует перемножить числа A и B, и ничего другого!

Это уровень понимания имеет место при поиске документальной информации, когда по тексту запроса информационно-поисковая система выдает потребителю все соответствующие этому запросу документы (такие документы называют релевантными, т. е. формально соответствующими запросу). Как видно, смысл сообщения на этом первично-семантическом уровне воспринимается вне контекста. Пример с запросом в информационно-поисковой системе наглядно показывает недостаточность первично-семантического уровня понимания для организации удовлетворяющего нас диалога. Действительно, совокупность релевантных документов обычно не совпадает с совокупностью документов фактически нужных, их называют пертинентными. К тому же некоторые пертинентные документы могут совсем не оказаться среди релевантных. Это будут потери информационно-поисковой системы, и их количество характеризует неполноту ответа. А с другой стороны, многие релевантные документы окажутся ненужными пользователю (потребителю). Это так называемый шум, который создается поисковой системой. Потери и шум в информационно-поисковой системе являются резуль-

татом того, что эта система в отличие от хорошего библиографа работает на первично-семантическом уровне, пренебрегающем контекстом запроса.

Так, на уровне семантики стихи Пушкина «Я вас любил: любовь еще, быть может, в душе моей угасла не совсем ...» могут быть поняты как выражение легкой грусти о прошедшей любви. Однако содержание их гораздо глубже. Это стихи о неувядающей Любви.

Третий уровень понимания

Итак, мы пришли к необходимости следующего уровня понимания — уровня *глубинной семантики.* Понимание текста на уровне глубинной семантики означает, что текст воспринимается как своеобразный наказ или «притча». Адресат видит в тексте некую даль, некий глубинный смысл, контекст, ради которого и появился данный текст. Понимание информационного запроса на глубинно-семантическом уровне означало бы понимание того, что на самом деле хочет получить потребитель информации. Очевидно, что такое понимание может быть достигнуто только в результате диалога, уточняющего цель запроса.

Понимание на глубинно-семантическом уровне упомянутого стихотворения А. С. Пушкина раскрывает в нем совсем иное, в некотором смысле противоположное тому, что замечено на уровне первичной семантики. Становится ясно, что это стихи о любви, не ослабевающей от разлуки, любви предельной, бескорыстной. Таким образом, этот уровень понимания основан на погружении общения в контекст. Общение строится на контексте и целиком опирается на него.

Вообще говоря, уровней понимания существует много (мы привели лишь основные), и зависят они от средств, которыми располагает компьютер. Очевидно, что переход с одного уровня понимания на другой связан с решением компьютером очень сложных задач (о них мы поговорим ниже). Но не следует думать, что в понимании можно ограничиться глубинно-семантическим уровнем, который отвечает разумным требованиям, предъявляемым к «понятливому» компьютеру. Можно пойти дальше и обнаружить ...

Четвертый уровень понимания

Говоря о рассматриваемых трех уровнях понимания, мы до сих пор считали, что слово «понимание» означает некоторое состояние, достигаемое субъектом, конечную цель процесса общения. Но почему бы не рассматривать понимание как открытый процесс, принципиально не имеющий завершения? На этом основании можно ввести и четвертый уровень понимания — *диалоги-*

ческий. На пример такого вида понимания обратил внимание М. М. Бахтин, который увидел особенность поэтики Ф. М. Достоевского как раз в диалогичности, открытости повествования, которое не завершается четкой авторской точкой зрения. Этим художественный мир Достоевского принципиально отличался от мира Л. Н. Толстого. У Толстого четкое деление героев на хороших, обладающих полнотой ощущения жизни, и дурных (точнее, неполноценных), живущих по искусственным правилам. Тем самым даже формально дурные поступки хороших героев оказываются оправданными, а хорошие поступки неполноценных ничего не стоят. У Достоевского принципиальное отсутствие завершенного понимания героя: они раскрываются в нескончаемом диалоге, и разделения на «овец и козлищ» так и не происходит. В самом дурном герое Достоевского в какой-то момент разворачивающегося диалога видится личность.

В диалогах Платона истина открывается в самом ходе диалога, и здесь также отсутствует завершенная философская система. Система знаний дается только в становлении, в процессе уточнения понятий. Это не дидактический прием, а принципиальный подход для выявления истины. В основе диалогического понимания всегда имеется какое-то противоречие, парадокс, действительный или кажущийся. Осознание противоречивости ситуации служит пружиной, разворачивающей диалог дальше. Таким образом, диалог устраняет имеющиеся противоречия и одновременно выявляет новые.

На диалогическом уровне участники общения сами формируют контекст, необходимый для углубления понимания. Разумеется, диалог в системе «человек — машина» нельзя по содержанию уподобить диалогам Достоевского или Платона. Но полезно иметь в виду, что высший уровень понимания возможен только в диалоге, именно этим он существенно отличается от более низких уровней понимания. Кроме того, диалогический уровень понимания не требует, чтобы оба собеседника находились на одном и том же уровне развития. Сократ в диалогах Платона ведет беседу с людьми скорее посредственными, но тексты этого диалога оказываются весьма поучительными и для нас, читателей.

Можно предположить, что человек все же сумеет достичь диалогического уровня понимания в диалоге с машиной, которая сама не выходит за рамки первичной семантики, но может обладать гигантским запасом фактических сведений. В процессе диалога человек сумеет эти факты осмыслить, и процесс общения перейдет на самый высокий уровень диалогического понимания. Во всяком случае эту возможность не следует заранее исключить из рассмотрения.

А теперь обратимся к проблеме понимания компьютером речи и предложений на естественном языке. Начнем с довольно забавной программы, имитирующей понимание.

ПРОГРАММА ЭЛИЗА

Эту программу создал американский ученый Дж. Вейценбаум и назвал ее так, вероятно, в честь Элизы Дулитл (героини известной пьесы Б. Шоу «Пигмалион», которую профессор фонетики Хиггинс из уличной торговки превратил в леди, обучив только правильному произношению). Программа ЭЛИЗА ведет диалог с собеседником — человеком, не понимая по сути дела ничего из сказанного им. Причем делает это она так хорошо, что неискушенный собеседник уверен, что ведет диалог с человеком,— настолько хорошо ЭЛИЗА имитирует диалоговое поведение людей. Заметим, что диалог ведется с помощью телетайпа — электрической пишущей машинки, подключенной к компьютеру с программой ЭЛИЗА.

Как же работает программа, которая так ловко водит за нос собеседника? Очень просто, так же, как человек, которому приходится поддерживать вежливую беседу со скучным собеседником на скучную тему. Он отделывается ничего не значащими репликами типа «А почему вы думаете, что ...?», или «Расскажите мне подробнее о ...», или «Доставляет ли вам удовольствие считать, что ...?» и т. д. Таких разговорных клише много. Они производят впечатление активности и заинтересованности. Но для использования таких клише необходимо заполнить многоточия, что-то поставить на их место. Выбор заполняющих слов и является самой сложной частью программы. Эти слова заимствуются из предыдущей реплики собеседника. Например:

С (собеседник): Не правда ли, сегодня хорошая погода?

Э (ЭЛИЗА): А почему вы спрашиваете?

С: Я не спрашиваю, а интересуюсь вашим мнением о сегодняшней погоде.

Э: Расскажите мне подробнее о сегодняшней погоде.

С: Нет дождя, светит солнце и можно гулять.

Э: А почему вы думаете, что можно гулять?

Читатель, наверное, заметил, как действует программа — она отыскивает ключевые слова в предложении и включает их в свою реплику-клише. Вот и все! Причем иногда это может быть невпопад, например «А почему вы думаете, что светит солнце?» Но собеседник, уверенный, что ведет диалог с человеком, а не с программой, думает, что тот рассеян и отвечает не слишком логично — с кем не бывает! Более того, некоторая нелогичность многими воспринимается как подтверждение, что диалог ведется с человеком, а не с машиной, которая славится своей «железной» логикой.

Особый успех ЭЛИЗА имела у одиноких людей, которым хочется пообщаться с кем-либо. Им не так важно, что именно говорит ЭЛИЗА, им важно, нужно самим высказываться. В одной из неврологических клиник США подключили ЭЛИЗУ к телефону и с

ней мог соединиться каждый желающий, но при условии, что диалог велся с помощью электрической машинки. Оказалось, что очень многие, «поговорившие» с ЭЛИЗОЙ были уверены, что общаются с внимательным и отзывчивым психотерапевтом — «Ведь он так хорошо понял меня!» И отказывались верить, что общались с компьютером.

Такая имитация понимания компьютером, естественно, понадобилась лишь для решения чисто научных задач, хотя и нашла свое применение в психотерапевтических целях. Но такой подход, конечно, не может быть использован для решения конкретных практических задач, возникающих при общении человека с компьютером. Здесь сразу возникает много трудностей, о которых стоит поговорить подробнее.

ТРУДНОСТИ ПОНИМАНИЯ СМЫСЛА ТЕКСТА

Все трудности понимания компьютером пользователя связаны с тем, что они «говорят» на разных языках. Компьютер понимает лишь язык машинных команд (о них мы говорили в гл. 3) или с помощью транслятора алгоритмические языки высокого уровня. Но в обоих случаях текст программы однозначно должен переводиться в машинные команды, исполнение которых приводит к решению поставленной задачи. Если же этого не происходит, то значит программа написана неправильно, если, разумеется, транслятор работал без ошибок.

Совсем иное происходит при общении с компьютером на естественном языке. В этом случае задача, поставленная пользователем, не решается именно потому, что нет правил построения «правильных» фраз естественного языка. Такими «правильными» для компьютера фразами следует считать фразы, которые понимаются им однозначно и не допускают нескольких толкований. Ведь компьютер не более чем автомат и его действия детерминируются его программой и поступающими данными, которые должны быть правильными.

Рассмотрим трудности понимания естественного языка, именно они препятствуют ЭВМ быть очень понятливым собеседником в диалоге с человеком.

Трудность первая

Она связана с многозначностью слов естественного языка. Например, русские слова «лук» и «коса» имеют больше одного значения. А английское очень распространенное слово set имеет более 1800 значений! Возникает очень трудная проблема — выбрать одно из нескольких значений. Для этого следует учитывать контекст. Например, чтобы выбрать одно из двух зна-

чений слова «лук», надо просмотреть его окружение и, если речь идет о еде, овощах, огороде и т. д., то его следует понимать одним образом, а если о спорте, оружии, стрелах и т. д. — другим образом. Хотя при этом вполне возможны ошибки. Так, легко ошибиться в понимании сцены свидания при косьбе двух молодых людей, среди которых одна девушка с косой.

Как видно, для правильного разрешения этих трудностей требуется вмешательство человека. Сделать это можно по-разному. Например, при компьютерном переводе с одного языка на другой многозначность может быть разрешена сразу — путем прямого обращения компьютера к человеку с требованием указать значение слова. Можно поступить иначе: в переводе компьютер приводит все значения многозначного слова из своей памяти, чтобы позже, на стадии редактирования этого текста, человек мог выбрать то, которое соответствует контексту. Этот последний способ широко используется в практике компьютерного перевода. Без человека преодолеть эту трудность нельзя.

Трудность вторая

Эта трудность связана со структурной неоднозначностью фраз естественного языка. Например, фразу «Он видел их семью своими глазами» компьютер может понять, что «он» имел семь глаз и видел «их». Здесь в самой фразе нет информации, является ли слово «семью» существительным (от слова «семья») или числительным (от слова «семь»). Для правильного понимания этой фразы компьютер должен знать, что «он» (если это человек) имеет не более двух глаз.

Здесь мы вплотную подходим к очень важной мысли: для правильного понимания человека компьютер должен иметь в своей памяти знания о том мире, в котором живут люди.

Трудность третья

Это неоднозначность «глубинной структуры» предложения. Видимая грамматическая структура однозначна, но допускает различные толкования смысла предложения. Например, фраза «Цыплята готовы к обеду» подразумевает, что кто-то кого-то скоро съест на обед. Но кто кого? Цыплята пообедают или их кто-то съест на обед? Здесь возможны оба варианта понимания. И только контекст поможет разобраться, кто кого съест.

Трудность четвертая

Она связана с так называемой семантической неоднозначностью, вызванной разной ролью одних и тех же слов в предложении. Например, в команде «Подать на пятый участок жел-

тый контейнер» непонятно, любой ли желтый контейнер или один из тех, о которых шла речь ранее, причем его «желтизна» лишь один из опеределяющих признаков этого контейнера (кроме этого, есть его номер, положение и т. д.).

Трудность пятая

Это так называемая «прагматическая неоднозначность», вызванная незнанием общеизвестных фактов. Например, переводя фразу «Она уронила карандаш на стол и сломала его», компьютер должен решить, что сломалось карандаш или стол. Ни сама фраза, ни контекст не помогут ему разрешить эту задачу. Для этого компьютер должен знать, что карандашом нельзя сломать стол. Другой пример. Фразу «Каша готова к обеду» люди понимают однозначно. Но компьютеру для этого нужно знать, что каша не живое существо, а еда. Для нас, предложение «Он стоял за забором в темной шляпе» не является двусмысленным: мы знаем, что заборы не носят шляп. Именно это должен знать компьютер для правильного понимания.

Как много мы должны заложить в память компьютера из того, что сами считаем само собой разумеющимся? Эти «знания» даже знаниями называть как-то неудобно. Но тем не менее это знания о нашем мире, и компьютер должен иметь эти знания, чтобы правильно понимать нас. Дело в том, что мы, люди, понимаем друг друга не только потому, что говорим на одном языке, но и потому, что живем в одном мире. А для того, чтобы компьютер (он не живет в нашем мире) понимал нас так же хорошо, как мы друг друга, нужно в его памяти построить модель нашего мира, т. е. сделать компьютер ... человеком (пусть пока в нечеловеческом обличии). Эту гигантскую по своей сложности и трудоемкости задачу едва ли удастся решить в ближайшем будущем. А пока? Неужели нет способов преодолеть такого рода трудности при общении с компьютером на естественном языке?

Именно здесь огромную роль играет диалог человека с компьютером, в процессе которого устраняется моногозначное толкование смысла фраз естественного языка. И очень может быть, что диалог как форма решения трудных задач останется еще надолго, а может быть, и навсегда (здесь под словом «трудный» подразумевается «неразрешимый формально»).

КАК КОМПЬЮТЕР ПЕРЕВОДИТ

Проблема перевода с одного языка на другой всегда была, остается и, по-видимому, еще долго будет актуальной проблемой современного общества с его огромными потоками разноязычной информации. И дело здесь вовсе не в трудностях перевода

художественных произведений — с ними должен справляться человек-переводчик (хотя компьютер и сейчас может помочь ему отыскивать синонимы, антонимы, фразеологические обороты и другие подобные справочные данные). Основная трудность в организации оперативного перевода научно-технической информации, мощные потоки которой поступают из-за границы и нуждаются в быстром осмысливании инженерами, научными работниками, администраторами. Здесь очень важно сохранить смысл текста (возможно, в ущерб стилю · и выразительности).

А так как компьютер сейчас самостоятельно не может справиться с этой задачей из-за трудностей, рассмотренных выше, то приходится обращаться к человеку (даже к двум) — предварительному и окончательному редакторам.

На первой стадии текст вводится в память компьютера. Далее предварительный редактор просматривает этот текст на экране дисплея, обрабатывает его так, чтобы он был пригодным для машинного перевода — устраняет те самые трудности многозначности, которые мешают компьютеру однозначно перевести текст. Так, многозначное слово сопровождается его пояснением, например фразу «Дайте мне лук» он дополнит в скобках словом «овощ» или «оружие», которое позволит компьютеру сразу сделать адекватный перевод. Фразу «Она уронила карандаш на стол и сломала его» он дополнит пояснением «(карандаш)», а фразу «Каша готова к обеду» достаточно дополнить замечанием «(сварилась)», чтобы устранить двусмысленность. Неоднозначность понимания текста можно устранить, введя специальные знаки пунктуации. Например, слова «старые мужчины и женщины» достаточно разделить скобками «[старые мужчины] и женщины», чтобы дать понять компьютеру, что речь идет лишь о старых мужчинах. Размеченный таким образом текст поступает на машинный перевод, результат которого просматривает окончательный редактор. Он устраняет грубые ошибки перевода и «приглаживает» полученный текст.

Такова наиболее распространенная и продуктивная схема компьютерного перевода с одного языка на другой. Любопытно, что в этой схеме оба редактора не обязаны быть переводчиками. Предварительному редактору достаточно лишь знать язык первоисточника, а окончательному — лишь тот язык, на который происходит перевод. Особенно удобен этот способ при переводе текста на несколько языков, так как предварительное редактирование при этом производится однократно.

КАК КОМПЬЮТЕР ПОНИМАЕТ

Пусть все трудности, связанные с многозначностью естественного языка, преодолены. И тем не менее компьютеру остается преодолеть последнюю и основную трудность — понять, что

именно сообщил ему пользователь. Эту трудность компьютер преодолевает за несколько этапов. Рассмотрим их.

Первым этапом является введение информации в компьютер. Если это делается с пульта дисплея, то никаких трудностей на этом этапе компьютер не испытывает. Действительно, текст, который предстоит ему понять, побуквенно вводится в память компьютера, и он сразу приступает к следующему этапу. Но если информация вводится голосом, то компьютеру предстоит еще преобразовать произнесенные слова в текст. Сначала для этого нужно ввести речь в память компьютера. Это осуществляется путем преобразования изменяющегося тока микрофона в последовательность чисел, которые и запоминает компьютер. Такого рода «оцифровку» производит аналого-цифровой преобразователь (АЦП), который преобразует непрерывное (аналоговое) значение тока микрофона в последовательность чисел, определяющих значения этого тока, измеренные через малые промежутки времени (например, через одну тысячную долю секунды). Так, каждое слово кодируется примерно 1000 числами. Эти числа несут всю необходимую информацию о смысле того, что сказал пользователь компьютеру. Для выявления этого смысла прежде всего следует набор полученных чисел (это запись речи) преобразовать в текст.

Задачу преобразования речи в текст выполняет *фонетический анализатор*. Это компьютерная программа преобразования чисел речи в буквы, слова и предложения текста. Такой анализатор должен решить ряд собственных задач; некоторые из них очень сложные. Например, задача разбиения произнесенной фразы на составляющие ее слова. Дело в том, что мы произносим слова без интервалов, слитно, и только знание этих слов позволяет слушателю понимать такую речь. Но компьютеру, чтобы узнать слова, из которых состоит фраза, нужно их сначала иметь. Иначе он не сможет воспользоваться словарем, в котором имеются значения этих слов. Получается порочный круг. Чтобы разбить слитную речь на слова, нужно знать эти слова, иметь ее уже в разделенном виде. Иначе, чтобы решить задачу, надо ее решить.

Эту трудность преодолевают очень громоздким образом — разбивают слитную фразу на отдельные слова каким-то приблизительным образом и проверяют, есть ли такие слова в словаре. И так до тех пор, пока не получится такое разделение на слова, которое будет впоследствии осмыслено.

Но тут встает другая, не менее сложная проблема «осмысливания» компьютером слов, произнесенных человеком. Ведь мы говорим совсем не так, как пишем. А компьютеру приходится искать в своем словаре слова, которых там нет, например «карову», «сонце», а так как все мы произносим слова по-разному, то можно представить, какую сложную задачу приходится решать фонетическому анализатору.

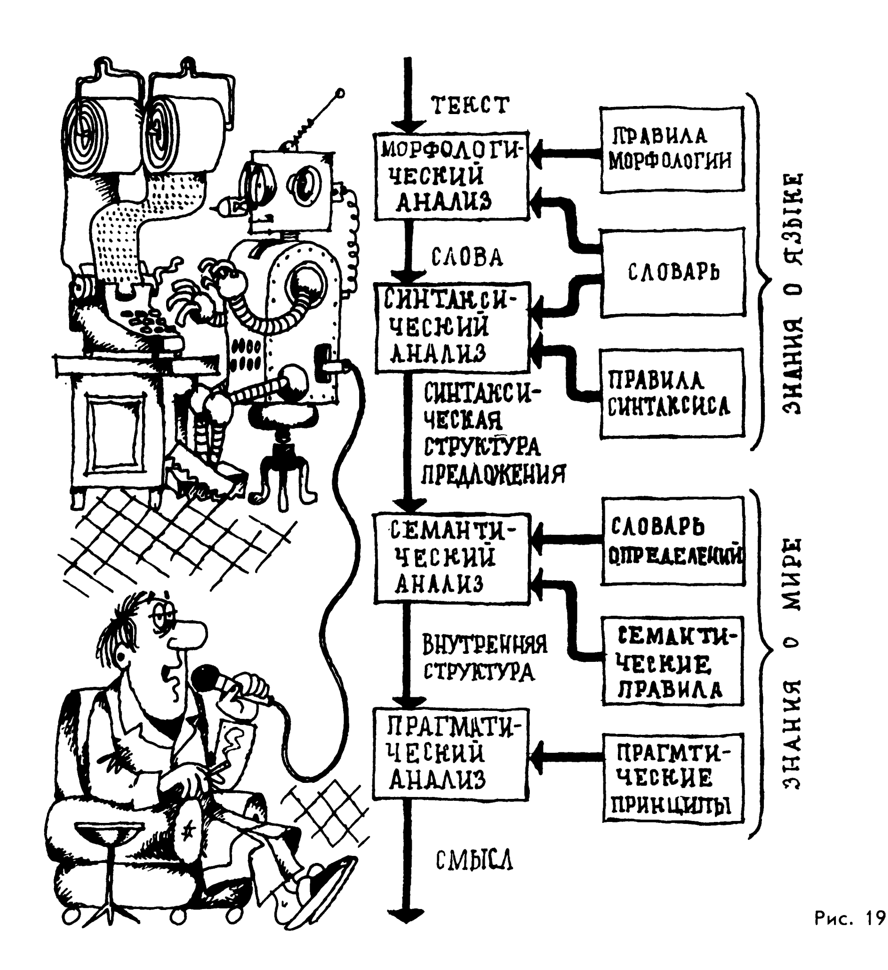


Рис. 19

Заметим, что мечта о таком анализаторе уже давно отразилась в легенде о фонетической пишущей машинке, которая могла бы печатать произносимый голосом текст. Выпускаемые и широко рекламируемые «голосовые пишущие машинки» позволяют фиксировать на бумаге только четко и внятно произносимые слова. Это автоматы для фонетического письма, которые пишут так, как говорится. Такое письмо нельзя показать никому: оно выглядит ужасно. Требуется тщательное редактирование, которое, как показывает опыт, занимает не меньше времени, чем написание текста «старым способом», вручную. Основная трудность, которую необходимо преодолеть для создания качественной фонетической пишущей машинки,— понимание компьютером произносимого текста. Поэтому пропустим стадию фонетического анализа и будем считать, что письменный текст не слитный и грам-

матически (точнее, морфологически) правильно записан в память компьютера. Причем многозначности в нем нет, она устранена на стадии предварительного редактирования текста.

Схема этапов решения задачи понимания компьютером и требуемая для этого информация показаны на рис. 19.

Первым этапом, с которого начинается анализ текста, является ...

Морфологический анализ

Это обработка слов, а точнее словоформ (так называют отрезок текста между двумя пробелами) текста как таковых, вне связи с контекстом. Каждому слову текста при морфологическом анализе ставится в соответствие морфологическая информация — это слово в единственном числе и именительном падеже, род и число слова и т. д. Например, словоформа «столов» будет иметь следующую морфологическую информацию: стол (это единственное число именительный падеж словоформы «столов»), существительное, мужской род, множественное число, родительный падеж, неодушевленное (отвечает на вопрос «что?»), нарицательное (в отличие от собственного имени).

Такой анализ можно осуществить декларативным и процедурным способом. При *декларативном* способе реализации морфологического анализа заранее составляется словарь всех возможных словоформ каждого слова с приписанной им морфологической информацией. При этом весь морфологический анализ делается заранее и хранится в виде соответствующего словаря в памяти компьютера. Задача декларативного морфологического анализа состоит только в поиске и отыскании нужной словоформы в словаре с последующим извлечением морфологической информации этой словоформы. При имеющихся компьютерных средствах хранения и поиска информации (базы данных и информационно-поисковые системы) такой способ «работает» очень быстро.

Но декларативный способ имеет один существенный недостаток: он требует большой памяти, ведь количество словоформ у одного слова очень велико. Так, у существительного их 12 (6 падежей для единственного и столько же для множественного числа), у прилагательного 36 (6 падежей, 2 числа и 3 рода), а у глаголов до сотни, с учетом отглагольных форм (3 времени: настоящее и будущее изменяются по лицам и числам, прошедшее — по числам, в единственном — по родам, причем отглагольная форма — причастие — изменяется по родам, числам и падежам). Именно поэтому словарь словоформ даже в узкой предметной области очень велик и может не вместиться в память компьютера (особенно персонального). Именно это и заставляет обращаться к *процедурному* морфологическому анализу. Этот

способ анализа действует в соответствии с правилами, которые мы изучали в 4—6 классах, когда учили морфологию русского языка. Эти правила легко формализуются и используются в процессе компьютерного морфологического анализа. Одноко при этом почти всегда возникает многовариантность, которую необходимо преодолевать на следующих этапах. Например, словоформа «территории» относится к родительному, дательному, предложному падежам единственного числа и к именительному и винительному падежам множественного числа. Но если перед данной словоформой стоит предлог «на» (он требует предложного падежа), то из указанных возможных вариантов остается лишь один.

За морфологическим следует ...

Синтаксический анализ

Этот анализ позволяет определить структуру предложения, опираясь только на результаты морфологического анализа. Здесь выявляются синтаксические связи между словоформами (точнее, между порциями морфологической информации, сопровождающими каждое слово). На стадии синтаксического анализа описываются все возможные языковые структуры предложения, исходя из правил грамматики (точнее, синтаксиса). Это тот самый разбор предложения, которому обучали нас в 7—8 классах. При этом исходным материалом являются лишь результаты морфологического анализа, т. е. очень незначительная информация. Вспомним, что школьник, делая разбор предложения, опирается прежде всего на его содержание, смысл, т. е. понимание того, какая ситуация описана в предложении. Это знание значительно облегчает синтаксический анализ. Например, при определении подлежащего мы ищем того (или то), кто (или что) производит действие, опираясь прежде всего на понимание того, что описано в предложении. Компьютер лишен этой возможности и вынужден действовать иначе. А чтобы добраться до понимания предложения (а именно оно является его конечной целью), он должен пройти этап синтаксического анализа, используя только результаты морфологического анализа словоформ этого предложения. Скудость имеющейся морфологической информации приводит к тому, что почти всегда появляется много вариантов допустимой структуры предложения, т. е. какие именно слова называть подлежащим, сказуемым и т. д. Эта неопределенность порождена именно отсутствием необходимых сведений. Например, в фразе «Толкатель движет деталь» после синтаксического анализа подлежащим может считаться и «толкатель» и «деталь», так как в русском языке подлежащее может находиться на любом месте (ср. «Деталь делает слесарь»).

Все полученные при синтаксическом анализе альтернативные синтаксические структуры предложения поступают на блок семантического анализа.

Семантический анализ

Этот этап обработки связан с выявлением смысла (семантики) предложения. Семантический анализатор преобразует синтаксическую форму предложения в «логическую» — в форму, понятную компьютеру. Наше человеческое понимание фразы мы можем проверить очень легко — по ответам на вопросы. Например, понимание предложения «Толкатель движет деталь» легко проверяется вопросами: «Что движет деталь?», «Что делает толкатель с деталью?» и «Что движет толкатель?» Человек, правильно отвечающий на эти вопросы, понял смысл этого предложения. При этом он опирается на свои знания о тех понятиях и действиях, которые использованы в предложении. Например, что толкатель может что-то двигать, а деталь может двигаться и это может быть нужно для чего-то. Такого рода знания необходимы для всякого понимания, ведь оно не что иное, как установление связи анализируемого текста с тем, что знает субъект. Если такой связи не устанавливается, то субъект ничего не понимает, да и в принципе понять не может.

Для компьютера понимание предложения связано с установлением связи слов этого предложения с понятиями, хранящимися в его памяти, точнее, с определениями этих понятий. Именно для этого нужен словарь определений. Этот словарь является компьютерным знанием о внешней среде, необходимым для понимания. Чаще всего этот словарь называют базой знаний (по аналогии с базой данных), реализована она может быть по-разному (как именно представляются знания в компьютере, мы расскажем в гл. 12).

Но понимания каждого из предложений, описывающих ситуацию в тексте, вовсе недостаточно для понимания этого текста. Ведь каждое предложение включено в некоторое окружение — контекст, выявлением которого занимается ...

Прагматический анализ

Он направлен на обработку связного текста. Одна из основных задач прагматического анализа — выявление взаимосвязи между предложениями текста. Типичный пример такой связи реализуется ссылками в виде личных местоимений: «Товары были отправлены на склад. Но он оказался переполненным». Здесь местоимение «он» обозначает «склад» (тот самый, на который были отправлены товары), что следует лишь из совместного анализа обоих предложений.

Особенно важен прагматический анализ для понимания в диалоге, который всегда имеет значительную контекстную зависимость. Рассмотрим, например, простой диалог:

А: Возьмите четыре красных кубика.

Б: Взял.

А: Сложите из них башню.

Здесь употребление ссылки «них» необходимо и не может быть заменено на «кубиков», иначе будет непонятно, из каких именно кубиков строить башню — взятых четырех красных или любых других.

В этом же примере диалога есть и другая трудность, которая должна быть понята: кто и что именно «взял» в реплике Б. Из контекста ясно, что именно Б взял «четыре красных кубика». Но здесь нет никакой ссылки, например в виде местоимения «их», и нет слова «я», они подразумеваются. Такого рода конструкции, в которых нет явно некоторых элементов, но которые восстанавливаются из контекста, называют неполными предложениями. Их очень много в связном тексте и еще больше в диалоге.

Для преодоления этих трудностей на стадии выявления смысла, следующего из анализа контекста, и нужны прагматические правила.

Метафоры — арбузная корка компьютера

По завершении прагматического анализа считается, что текст полностью понят компьютером, хотя это иногда бывает совсем не так. Например, компьютеру пока «не по зубам» понять метафоры, столь естественные в речи человека. И дело здесь вовсе не в поэзии, которая насквозь пронизана метафорами (бог с ней, поэзией, не так уж важно, чтобы компьютер понимал правильно стихи). Дело в том, что метафоры очень часто используются в человеческом общении на весьма прозаические темы и повседневная речь насыщена ими. Например, «надо провертеть эту идею» или «я затратил целый день, чтобы разложить свои соображения по полочкам». Чтобы компьютер правильно понял эти фразы, в его базе знаний должны быть заложены расшифровки оборотов о «верчении идеи» и «раскладке по полочкам». В этом отличие компьютера от человека. Он нуждается в таких расшифровках, а человек догадывается о них. Например, выражение «пудрить мозги» понятно всякому, даже тому, кто его раньше и не слышал. Компьютеру же это не под силу, если такого выражения нет в его базе знаний. Не поймет он и названия этого раздела, если в его базе знаний не найдется расшифровки выражения «поскользнуться на арбузной корке».

ЧТО ЖЕ ДАЛЬШЕ?

В силу указанных трудностей хороших программ понимания естественного языка очень мало. Все они ориентированы на узкую предметную область. Оно и понятно. Чем у́же предмет-

ная область, в рамках которой происходит диалог с компьютером, тем проще создать достаточно полную базу знаний этой области. И, следовательно, тем точнее может быть понимание компьютером фраз из этой предметной области.

Но при создании компьютерных систем понимания естественной речи есть трудности принципиального характера. Такая система, чтобы быть эффективной, должна знать не только что она знает, но и суметь объяснить пользователю, что и почему она не понимает. Но такого рода умение характеризует не что иное, как *самосознание* системы.

Мы еще мало знаем о феномене самосознания и поэтому пока не умеем его воспроизводить в необходимой полноте. Но первые шаги в этом направлении уже делаются (о них мы расскажем в гл. 14, где описаны экспертные системы, обладающие элементами самосознания в виде подсистемы объяснения, как система пришла к данному полученному выводу). Необходимость самосознания — не столько следствие языка общения, сколько свидетельство сложности нашего мира, относительно которого происходит общение. Что же делать, пока компьютер не способен к эффективному самосознанию? Идти ему навстречу и не ставить его в затруднительное положение. Для этого достаточно общаться с ним простыми фразами, без метафор, с минимальным контекстом и в узкой предметной области. Такой упрощенный естественный язык вполне по силам понять современному компьютеру, точнее, его программе понимания естественного языка. Поэтому заканчивает эту главу...

ГИМН КАНЦЕЛЯРИТУ

Именно на таком языке пишется деловая документация (письма, инструкции, отчеты, заявления и т. д.). Этот язык неоднократно осужден литературоведами, назван канцеляритом и проклят критиками и любителями изящной словесности. Но он выжил. Дело в том, что канцелярит нужен. Он выражает извечное стремление человечества к правильному пониманию, когда дело доходит до серьезных вещей, не допускающих двусмысленности толкования. Например, описание технологии изготовления детали должно быть одинаково понято каждым, кто имеет отношение к изготовлению этой детали, — проектировщиком, технологом, мастером, рабочим.

Очевидно, что такой язык (его сейчас называют языком деловой прозы) открывает реальные возможности для эффективного общения человека с компьютером. Его выразительности вполне достаточно для того, чтобы нам вступить в диалог с машиной по практически важным поводам, связанным с человеческой деятельностью в области науки, техники, производства и

т.д. Задушевного разговора на канцелярите не получится, да он и не нужен, пока.

— Что же получается,— задумчиво заметил Мегрэ,— сначала столько разговоров о неограниченных возможностях современного компьютера, а потом автор идет на попятную. Оказывается, компьютеру не по зубам самые простые вещи, которые запросто делает любой человек. Ведь что может быть проще, чем общение на естественном языке?

— Здесь не так все просто, как кажется,— улыбнулся Поль.— Во-первых, не всегда просто и естественно общение даже между людьми. Вспомните, как трудно объяснить что-либо незнакомому человеку. И все потому, что вы не знаете, что знает и чего не знает он. Поэтому, чтобы понять друг друга, важно много знать друг о друге. И чем больше это знание, тем лучше понимание. Вы ведь по-разному отдаете распоряжения мне и, например, новичку Жаку.

— Выходит, чтобы хорошо понимать меня, компьютер тоже должен съесть со мной пуд соли?— ехидно заметил Мегрэ.

— Увы, месье, это так. Хотя сделать это сможет далеко не всякий компьютер. Для этого он должен иметь возможность самообучения — запоминать и учитывать в дальнейшем специфику каждого собеседника. Чем больше такой компьютер общается с вами, тем лучше он вас понимает, точно так же, как при общении людей. Так что без «пуда соли» здесь не обойтись.

— Это уже что-то из области научной фантастики. Не так ли, Поль?

— К сожалению, таких компьютеров, точнее таких программ общения с человеком, пока нет. И думаю, что они появятся нескоро. Человек легко адаптируется и, общаясь с компьютером, который его плохо понимает, начинает приспосабливаться к нему. Важно, чтобы между человеком и компьютером поддерживался активный диалог, в процессе которого и появится возможность взаимной адаптации. Человеку это сделать проще: он делает это каждый день, общаясь с разными людьми и приспосабливаясь к каждому из них. Точно так же он легко адаптируется к компьютеру.

— Выходит, не нужен и хороший компьютер — человек всегда приспособится к нему,— заметил Мегрэ.

— Конечно, нет,— улыбнулся Поль.— Если для общения приходится затрачивать слишком много усилий, то проще отказаться от него. Так часто мы и поступаем, например при общении на малознакомом нам языке с иностранцами. Здесь толкового общения никогда не будет при всем желании иностранца адаптироваться к нашему полузнанию языка. Так и компьютеру, не имеющему хоть какую-то программу общения на естественном языке, никогда не растолковать своей задачи без алгоритмических языков. Поэтому чем выше уровень решаемых задач, тем сложнее должны быть программы общения с компьютером. Адаптивность человека в таком диалоге лишь подспорье, но не решает задачи общения.

— Что ж, как видно для решения самых сложных задач придется ждать сверхсовершенных программ. Не проще ли их решать без компьютера?

— А мы так и поступаем. Остается довольно много областей человеческой деятельности, пока недоступных компьютеру. Это политика, этика, мораль, искусство и другие неформализованные области. Так что поговорить с компьюте-

ром на такие темы и тем более поручать ему решать проблемы из этих областей нам удастся не скоро.

— Ну что ж,— добродушно сказал Мегрэ,— это обнадеживает. Я могу легко делать то, что не по зубам самым совершенным компьютерам!

10. ОТЦЫ И ДЕТИ, ВНУКИ И ПРАВНУКИ

(Компьютеры пятого поколения)

ТЕМП, ТЕМП, ТЕМП...

Компьютеры, как и люди, имеют свои поколения. И, как у людей, каждое новое поколение в чем-то отрицает старое. Но смена поколений компьютеров происходит значительно быстрее — примерно каждые 10 лет, причем новое поколение отличается от старого (по производительности, емкости памяти, стоимости, габаритным размерам и т. д.) примерно в 10 раз. Такой огромный темп развития еще не имела ни одна отрасль промышленности нигде и никогда (напомним, что эра компьютеров началась лишь в 1945 г.).

Так, процессор первых машин едва размещался в огромном шкафу (а то и в нескольких). Сейчас же процессор размещается в объеме, значительно меньшем спичечного коробка. Это меньше более чем в миллион раз по сравнению с первыми ЭВМ. При этом производительность процессора возросла более чем в тысячу раз — с тысячи операций в секунду в первых компьютерах до миллиона в коммерческих процессорах, а в специальных — значительно больше. Одновременно стоимость процессора сейчас снизилась до нескольких рублей. Аналогичны соотношения по емкости памяти, надежности и т. д. И все это произошло всего за 40 лет, при жизни одного поколения людей. Темпы, как видим, небывалые!

Если бы таким стремительным темпом только 25 лет развивалось самолетостроение, то самолет типа нашего аэробуса (ИЛ-86) стоил бы всего 500 руб., совершал облет земного шара за 20 мин и тратил бы при этом 20 л горючего!

Очевидно, что при столь ошеломляющих успехах развития компьютерной техники следует ожидать, что она вскоре дойдет до каждого из нас. Этот момент, грубо говоря, и будет соответствовать пятому поколению компьютеров. Ожидают его к 2000 г., но занимаются пятым поколением, и весьма интенсивно, уже сейчас. А пока «на дворе» третье и четвертое поколения компьютеров.

С ЧЕГО ВСЕ НАЧАЛОСЬ

Идея компьютеров пятого поколения впервые возникла в Японии в 1979 г. и была воспринята всеми как фантастика — уж очень смелыми были предложения и предположения о компьютерах ближайшего будущего. Но эти сомнения были рассеяны, когда был опубликован Японский национальный проект создания машин пятого поколения. В соответствии с этим проектом к 1990 г. должны быть созданы первые образцы таких машин, а к началу второго тысячелетия компьютеры должны войти в каждый дом, как телевизор, магнитофон или стиральная машина, должны стать такими же простыми в обращении и столь же доступными по стоимости.

Сама по себе идея всеобщей компьютеризации, заложенная в проект машин пятого поколения, не нова и вполне реальна, если для этого воспользоваться персональными компьютерами (о них мы уже говорили подробно в гл. 2). Но обеспечить каждого из нас индивидуальным компьютером — еще не значит обеспечить всех всей необходимой информацией и средствами ее переработки.

Дело в том, что персональный компьютер создает очень удобную, но лишь локальную компьютерную среду. Но эта локальность становится непреодолимым препятствием, если необходимо оперативно получить информацию, которая содержится в удаленном от вас банке данных, или необходимо решить задачу, для которой потребуется вычислительная мощность, значительно превышающая возможности вашего персонального компьютера. Эту проблему позволяют решить вычислительные сети (о них было рассказано в гл. 7). Такого рода сети просто необходимы для компьютеров пятого поколения. Но и сам персональный компьютер — не идеальный партнер, общение с ним требует определенных навыков, компьютерной грамотности. Очевидно, что для эффективного общения с компьютером должен быть налажен речевой контакт, иначе он никогда не станет нашим надежным партнером и собеседником.

Все это и многое другое и вызвало необходимость создания компьютеров пятого поколения. Обеспечение каждого пользователя простой, надежной и оперативной возможностью пользоваться огромным вычислительным потенциалом и практически неограниченной информацией — основная идея этого поколения. А для ее реализации необходимо разработать новые технические средства, новые методы программирования и новые способы общения с компьютером. Решение этих задач и обеспечит человечество новыми средствами обработки информации — компьютерами пятого поколения.

Новизна этих средств состоит еще и в том, что с ними мы вступаем в новый этап развития ЭВМ — этап компьютерных умозаключений. Напомним, что первым этапом был вычислительный,

откуда и произошли имена «ЭВМ» и «компьютер», а вторым — общеинформационный, когда компьютер стал применяться во всех областях обработки информации, а не только для вычислений. На новом (пятом) этапе компьютер вторгается в интеллектуальную деятельность человека, позволяя автоматизировать процесс умозаключений, что открывает новые возможности для решения интеллектуальных задач.

А теперь рассмотрим подробнее основные черты компьютеров пятого поколения. Начнем с того, что ответим на вопрос: почему именно пятое поколение? И какими были предыдущие?

ПОКОЛЕНИЯ КОМПЬЮТЕРОВ

Развитие компьютерной техники подчиняется законам развития техники вообще. Темп развития любой отрасли определяется ее технологией. Чем совершеннее технология, тем выше темп развития этой отрасли, и наоборот. Так вот, в компьютерной технике новая технология появляется чрезвычайно интенсивно. Именно это обстоятельство обеспечивает столь высокие темпы развития вычислительной техники.

Каждое новое поколение компьютеров отличается от предыдущего элементной базой, т. е. теми «кирпичиками», из которых состоит компьютер.

Первое поколение компьютеров

В самом начале, в первом поколении, таким элементом была электронная лампа, в которой использовался так называемый эффект Эдисона. Великий изобретатель сделал это открытие между делом (а делом он считал изобретательство). Любопытно, что Эдисон, патентовавший все свои изобретения, это открытие не запатентовал: он не увидел в нем никакой практической пользы. Но именно это открытие Эдисона дало первый толчок развитию вычислительной техники, который в конечном счете привел к триумфальному шествию компьютеров. В то время электронные лампы Эдисона широко использовались в радиоаппаратуре. И именно к ним обратились создатели первых ЭВМ.

Машины первого поколения размещались в огромных залах (типа спортивных). Тысячи электронных ламп быстро нагревали помещение, высокая температура создавала не только неудобства для обслуживающего персонала, но и снижала надежность работы самой ЭВМ, и отвод тепла стал одной из самых острых проблем организации эффективной работы такой машины. Так электронная лампа, благодаря которой начался компьютерный бум, стала на пути прогресса вычислительной техники. Ее срочно нужно было заменить на другой элемент, более «холодный», экономичный и меньших габаритов. Такой элемент был найден.

Им оказались полупроводниковые элементы на базе кристаллов кремния, германия, галлия, селена, а также арсенида галлия, карбида кремния и т. д. Эти материалы имеют различную проводимость в различных направлениях кристалла (в отличие от электронной лампы, имеющей обратное сопротивление, близкое к бесконечности, полупроводниковые диоды имеют лишь разное сопротивление в обоих направлениях, отличающееся в 100—10 раз). И такие полупроводниковые элементы немедленно были использованы для компьютеров. Так началось...

Второе поколение компьютеров

Миниатюрный транзистор (так стали называть полупроводниковый элемент) был значительно совершеннее электронной лампы, что сразу сказалось на свойствах компьютеров и привело в действие закон «10»— улучшение за 10 лет всех характеристик компьютера примерно в 10 и более раз. Машины второго поколения обладали настолько высокими характеристиками, что некоторые успешно эксплуатируются до сих пор (машины первого поколения можно встретить разве лишь в музеях истории техники). Например, и сейчас работает наша великолепная машина второго поколения БЭСМ-6. Она имеет производительность миллион операций в секунду, чего не всегда достигают коммерческие машины третьего и четвертого поколений. Но по мере развития вычислительной техники, требующей все более и более компактных решений, полупроводниковые приборы стали тормозить процесс повышения эффективности машин. Стремление к компактности электронных схем компьютера связано не только с желанием сделать его поменьше (хотя его габаритные размеры иногда решают успех его применения, например на борту самолета или в космосе). Компактность нужна прежде всего для повышения производительности процессора и его надежности. Длинные проводники, связывающие полупроводниковые элементы, естественно, задерживали процесс распространения сигнала по схеме и тем самым снижали ее быстродействие. Быстрая схема должна быть микроскопических размеров. Так возникла мысль о микроэлектронном исполнении схем. Но решила дело микроэлектронная технология. Именно она породила...

Третье поколение компьютеров

В этом поколении элементную базу компьютеров образовали так называемые интегральные схемы. Замечательное отличие такой схемы заключается в том, что все ее элементы (транзисторы, резисторы и конденсаторы) и соединения между ними создаются на небольшой пластине кристалла (обычно кремния) площадью порядка 1 $см^2$. Технология, использующая процессы

травления и напыления, позволяет создавать схемы с чрезвычайно мелкими элементами. Именно поэтому такие схемы и стали называть интегральными микросхемами. Очевидно, что совершенствование технологии создания микросхем позволяет «уложить» на единице площади все больше элементов и соединений между ними. Именно технология определила дальнейшее развитие вычислительной техники и следующие поколения компьютеров. Пока технология была еще несовершенна, микросхемы имели малую степень интеграции — порядка 10 транзисторов на схему. Микросхемы средней степени интеграции (СИС) имеют до 100 транзисторов на схему. С совершенствованием технологии появились большие интегральные схемы (БИС), имеющие до 1000 транзисторов в одной схеме. Именно они дали жизнь третьему поколению компьютеров. Примером машин этого поколения являются известные и самые распространенные у нас машины серий ЕС (Ряд-1) и СМ.

Четвертое поколение компьютеров

Дальнейшее совершенствование технологии позволило создать сверхбольшие интегральные схемы (СБИС), содержащие 100 тыс. транзисторов и более. Именно СБИС стали основой элементной базы компьютеров четвертого поколения. На такой схеме реализованы и память компьютера, и арифметическо-логическое устройство (АЛУ) и даже процессор. Процессор, реализованный на одной СБИС, получил название микропроцессора. Появился он сравнительно недавно, в 1971 г. (это микропроцессор Intel 4004 с 2250 транзисторами), и сразу стал вехой в развитии компьютерной техники.

«Сердцем» каждого компьютера является процессор (напомним, что он объединяет АЛУ и устройство управления). Появление микропроцессорного «сердца» открыло перед создателями компьютеров огромные возможности. Теперь даже радиолюбитель может собрать собственный персональный компьютер, добавив оперативную память (на БИС или СБИС) и устройства ввода-вывода (на нескольких СИС). Это обеспечила микроэлектронная технология!

И именно прогресс микроэлектронной технологии открыл реальный путь к созданию компьютеров пятого поколения. В этом поколении элементной базой являются микропроцессоры. Компьютеры пятого поколения будут создаваться на базе микропроцессоров, так же, как компьютеры первого поколения на лампах, а второго — на транзисторах. Известно, что микропроцессор позволяет реализовать любую функцию переработки информации, что и открывает перед такой элементной базой почти неограниченные возможности по созданию компьютеров для решения сложнейших задач, выдвигаемых современным производством, экономи-

кой, наукой, и ... каждым из нас. Эти последние задачи, как это ни покажется странным, станут одними из основных для компьютеров пятого поколения.

Требования к компьютерам пятого поколения

Мы привыкли, что компьютерная техника обслуживает преимущественно науку, промышленность, народное хозяйство и т. д., а задачи, которые волнуют лично каждого из нас, приходится решать нам самим. И дело здесь вовсе не в том, что мы не можем «выйти» на ЭВМ — сейчас их достаточно. Просто большинство из нас не будет знать, что делать с компьютером, даже если он появится в вашей квартире — уж очень много нужно знать, чтобы обращаться с компьютером для решения своих конкретных задач как на работе, так и дома.

Чтобы преодолеть этот барьер, прежде всего необходимо, чтобы компьютер мог понимать естественный язык, чтобы общение с ним было не сложнее, чем с коллегой по работе или домочадцами. Именно таким должен быть компьютер пятого поколения.

Но чтобы понимать естественный язык (текст или речь), компьютеру нужно очень многое знать и уметь. Для этого он должен иметь чрезвычайно большую оперативную память и огромное быстродействие. Именно для этого используются СБИС. Компьютер пятого поколения должен иметь быстродействие 10^{11}—10^{12} операций в секунду — триллион! Это в миллион раз больше, чем у нынешнего среднего компьютера! Для чего же нужна такая производительность? Прежде всего, чтобы быстро манипулировать своими знаниями, чтобы сразу реагировать в процессе общения с человеком и с другими компьютерами. А чтобы обеспечить такое быстродействие, очевидно, недостаточно одного процессора, производительность которого обычно не превышает 1 млн. операций в секунду. Поэтому компьютер пятого поколения будет многопроцессорной системой, широко использующей распараллеливание процесса обработки информации (о способах распараллеливания мы говорили в гл. 6). Для этого в компьютерах пятого поколения в основном будет использоваться метод потока данных. Этот метод отличается от традиционной обработки, в которой вычислительный процесс управляется потоком команд программы (фоннеймановская архитектура, основанная на однопроцессорном компьютере). При многопроцессорной схеме компьютера логично поступать по-другому: не команды, а данные сделать управляющими вычислительным процессом. Именно в этом и состоит метод потока данных.

Напомним, что реализуется он следующим образом. Как только появляются данные для выполнения какой-то части программы (эти данные могли появиться извне или получены другой частью программы) и есть свободный процессор, он сразу загружа-

ется этой программой. Поэтому при работе по методу потока данных процессоры простаивают мало, что и обеспечивает эффективное распараллеливание решения задачи.

Но, как легко вычислить, чтобы создать компьютер триллионной производительности, нужно не менее миллиона процессоров-миллионников. Даже для СБИС это слишком много! Поэтому следует искать пути убыстрения вычислений, кроме распараллеливания. И такой путь имеется — создание специальных СБИС, обрабатывающих информацию не программно, а аппаратно — так, как это делается в карманном калькуляторе. Например, операция умножения обычно выполняется программно (точнее, микропрограммно), но можно сделать специальную схему — умножитель, которая выполнит эту операцию, естественно, значительно быстрее (именно так поступают при создании высокопроизводительных компьютеров).

Переход на схемное решение значительно увеличивает скорость решения задачи (точнее, подзадачи), иногда на несколько порядков. Именно поэтому в компьютерах пятого поколения должна широко использоваться схемная обработка или, как иногда говорят, программирование на кристалле (имеется в виду кристалл интегральной микросхемы) или кремниевое программирование (микросхема обычно изготовляется на кристалле кремния). Но очевидно, что создавать такой специальный запрограммированный кристалл (или заказную СБИС) следует только для типовых и широко используемых программ обработки, так как его проектирование и изготовление — очень трудоемкая процедура. Но чем больше таких различных СБИС будет использовано в компьютере, тем выше станет его производительность. Именно эта идея закладывается в компьютеры пятого поколения, что требует дальнейшего развития технологии СБИС.

Но одной производительности для компьютера, который должен понимать и решать задачи, мало. Нужна огромная оперативная память, способная хранить знания, на основе которых действует компьютер. Для компьютеров пятого поколения нужна емкость памяти до 10^{12} байт (знаков) — столько имеют миллион томов книг по 400 страниц каждый. Это значит, что в оперативную память компьютера пятого поколения можно будет вложить большую городскую библиотеку, а чтобы перевести в оперативную память все хранилища библиотеки им. В. И. Ленина (одной из крупнейших в мире) понадобится не более 50 таких компьютеров. Но такая гигантская память должна быть очень хорошо организована, иначе быстро найти в ней ничего не удастся. Именно поэтому компьютеры пятого поколения будут хранить в своей оперативной памяти базу данных, организованную удобным образом. Одной из таких перспективных баз данных, которая планируется для использования компьютерами пятого поколения, является...

РЕЛЯЦИОННАЯ БАЗА ДАННЫХ

Это множество таблиц, каждая таблица выражает определенное отношение (англ. ralation) между строками и столбцами. Например, кадровая таблица определяет отношения между строками, где записаны данные по каждому работнику, и столбцами, определяющими его фамилию, год рождения, образование, зарплату и т. д. А библиографическая таблица устанавливает отношение между строками, где записаны библиографические данные, и столбцами, в которых фиксируется фамилия автора, название статьи или книги, издательство, год выпуска, объем публикации и т. д. Любой фрагмент такой таблицы тоже определяет отношение. Например, отношение между образованием и зарплатой легко определить из кадровой таблицы, вычеркивая все столбцы, кроме двух интересующих.

Реляционная база данных состоит из набора таких таблиц, представляющих некоторую предметную область, интересующую многих пользователей. Но хранение этих сведений в банке — не основная его функция. Банк данных создается ради его использования. По запросу пользователей информация, хранимая в банке, может быть выдана ему в требуемой форме, в виде определенной таблицы-отношения. Реляционная база данных позволяет манипулировать своими таблицами и создавать новые, интересующие пользователя. Например, выявить отношение между возрастом работника и числом его публикаций, для этого понадобятся две описанные таблицы. Все необходимые действия с таблицами выполнит сама база данных, точнее, система управления базы данных (СУБД).

Такого рода манипуляции с таблицами отношений осуществляются методами реляционной алгебры, которая позволяет выявлять нужные пользователю (или компьютеру) отношения между данными, хранящимися в базе. Именно реляционная база данных является основной формой хранения информации компьютеров пятого поколения.

Мы уже говорили о том, что тенденция развития компьютерной техники такова, что на компьютер возлагается все больше невычислительных операций. И вычислительная машина вычисляет (относительно) все меньше и меньше. А основное свое время она занята тем, что называется логическим выводом, который необходим для решения информационных задач. В процессе решения информационных задач компьютеру приходится все чаще и чаще обращаться к базе данных, чтобы на ее основе делать логические выводы. Примером такого рода вывода является выявление в базе данных определенного свойства у заданного объекта, если это свойство в базе упоминается не вместе с объектом, а через другие его свойства, как в известном силлогизме: все рыбы плавают, карась — рыба, следовательно, он плавает. Такого

рода логических выводов компьютерам пятого поколения придется делать очень много: их «знания» определяются содержимым банков данных и логический вывод является средством извлечения этих знаний из банков данных.

На один логический вывод компьютер обычно затрачивает 100—1000 операций. А так как таких выводов приходится делать очень много, то производительность компьютеров удобно исчислять количеством логических выводов в секунду. Так, компьютеры пятого поколения должны делать до 1 млрд. логических выводов. Очевидно, что для создания программ с большим числом логических выводов необходим специальный язык программирования, ориентированный на логические выводы. Такими языками являются Пролог (именно он был положен в основу японского проекта компьютеров пятого поколения) и Лисп (о них мы расскажем в гл. 13).

В заключение отметим, что здесь описаны некоторые технические средства, типичные для компьютеров пятого поколения. Именно с их помощью будут решаться задачи. Сами же задачи компьютеру будут ставиться словесно, так как общение с компьютерами пятого поколения будет происходить на естественном языке. Очевидно, что и методы решения таких задач должны быть новыми. Для компьютеров пятого поколения такими методами являются методы искусственного интеллекта.

— И все-таки непонятно, зачем иметь такие гигантские вычислительные мощности пятого поколения каждому пользователю,— заметил Мегрэ.— Я хорошо понимаю, что существуют суперзадачи, для которых нужно иметь суперкомпьютеры. Но таких задач не так много, а для многих нужд достаточно персонального компьютера с телефонной связью, чтобы получить необходимую информацию. Не заболели создатели компьютерной техники гигантоманией?

— Нет, разумеется,— улыбнулся Поль.— Эти мощности в пятом поколении компьютеров будут платой за сервис, предоставляемый пользователю. Этот сервис не совсем обычный (быстро, качественно, надежно), а *интеллектуальный*. Он обеспечивает прежде всего *понимание* пользователя компьютером. Конечной целью такого сервиса является «понимание с полуслова». Но для этого компьютер прежде всего должен понимать естественную речь неискушенного пользователя, переспрашивать его до тех пор, пока не убедится в правильности понимания. Более того. Очень часто мы нечетко знаем, чего хотим. Наши желания зачастую расплывчаты и неопределенны. В такой ситуации даже человек с трудом может понять, чего же хочет от него собеседник. Чего же вы хотите от компьютера?

— Я ничего не хочу!— поспешно сказал Мегрэ.— Это вы от него требуете то, чего и человеку-то не всегда под силу. Ну и пусть компьютер не понимает человека, когда тот сам себя не понимает. Пусть исполняет четкие распоряжения или, как это говорят математики, хорошо поставленные задачи. А плохо поставленные пусть решает сам. На то он и человек!

— Ваше деление задач на «хорошие» и «плохие» правильно, комиссар.

И все компьютеры до четвертого поколения включительно решали именно «хорошие» задачи. Но, к сожалению, хороших задач очень мало, именно поэтому компьютеры использовались не на каждом шагу. «Плохих» задач значительно больше, чем «хороших», да они и важнее. Наверное, именно поэтому самые ответственные задачи не доверялись компьютеру — их решали люди. Именно потребность (а не только желание) решать «плохие» задачи привела к появлению компьютеров пятого поколения. Эти задачи довольно четко делятся на два класса: понять человека и выполнить то, что было понято. Для решения этих задач нужно располагать огромной информацией и перерабатывать ее с огромной скоростью.

— А какая такая «огромная» информация нужна компьютеру для понимания человека?— спросил Мегрэ.

— Это прежде всего модель мира, в котором живет человек. Не располагая такой моделью, компьютер просто не может понять человека, если он общается с ним на естественном языке, А именно такое общение свойственно человеку. И никакая компьютерная грамотность не заставит его предпочесть формальный язык естественному. Кроме того, естественный язык — самый выразительный. Только на нем удается описать и сказать все, что необходимо. Формальные языки, и тем более языки программирования, выразительны только в той области, для которой они были созданы.

— Хорошо. Чтобы компьютер понял человека, действительно нужны и огромная память, и огромное быстродействие. Но нужны ли они для выполнения заданий. Чтобы ответить на вопрос: дважды два?, нужно лишь его понять и обратиться к таблице умножения, которая занимает невесть какую память,— заметил Мегрэ.

— Если бы человечество ограничивалось только такими вопросами!— грустно сказал Поль.— К сожалению (а может быть, к счастью), чаще задают другие вопросы.

— Например?

— «Как изменится климат в Азии, если повернуть на юг реки, текущие на север?» или «Каковы последствия введения налога на курящих?» и т. д. и т. п. Таких вопросов слишком много. А ответы на них требуют огромной исходной информации не только в виде данных, но и в виде моделей, связывающих эти данные. Эти самые модели являются по сути дела закономерностями нашего мира. Прежде чем отвечать на вопрос, надо либо найти их в каких-либо банках данных и знаний, либо выявить из имеющихся наблюдений. На все это нужны очень большие запасы информации и вычислительные мощности, которыми и будут располагать компьютеры пятого поколения.

— Ну хорошо, Поль. Я готов поверить, что все это очень нужно и даже необходимо для научного, промышленного и всякого другого прогресса. Но какую помощь эти компьютеры окажут нам, сыщикам? Только речевое общение с компьютером? Позволит сократить секретаршу, которая вводит данные в компьютер? Не мало ли?

— Пятое поколение позволит прежде всего решать задачи, которые мы сейчас решаем плохо и иногда вообще не можем решить. Вот эту гору данных,— Поль указал на стопы сообщений, полученных на запросы Мегрэ со всего света,— сможет обработать и дать вам ожидаемый ответ только машина пятого поко-

ления, в памяти которой будут находится известные взаимосвязи преступного мира.

— Что ж, спасибо на светлом будущем,— сказал Мегрэ и поспешно добавил: — Не обижайтесь, Поль, просто хочется уже сейчас воспользоваться этими будущими компьютерами пятого поколения. Они мне вот так нужны! — и он провел ладонью по горлу.

— Вот такие потребности и движут развитие компьютерной техники,— улыбнулся Поль.

11. СУРРОГАТ ИНТЕЛЛЕКТА

(Искусственный интеллект)

СПЕЦИФИКА МЕТОДОВ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Понятно, что компьютеры пятого поколения должны работать по-новому, на новых принципах обработки информации. Дело в том, что в отличие от компьютеров предыдущих поколений, им придется иметь дело с очень сырой исходной информацией. Это прежде всего речь, текст, изображения... Такого рода информация плохо организована или, как говорят строго, плохо структурирована. А для функционирования «нормального» компьютера требуется, чтобы исходные данные были хорошо структурированы. Именно поэтому и необходимо создать методы, позволяющие компьютеру работать в такой «непривычной» для него обстановке.

Такими методами стали методы искусственного интеллекта. Так была названа новая область научных исследований, которая к человеческому интеллекту имеет довольно косвенное отношение. Речь идет о задачах, которые очень хорошо решаются человеком и плохо вычислительной машиной. Можно было бы не тужить о трудностях компьютера (ведь человек-то справляется с этими задачами хорошо), если бы такие задачи не стали встречаться столь часто, что их решение нуждалось в автоматизации, хотя бы из экономических соображений.

Естественно, далеко не все задачи, хорошо решаемые человеком, следует относить к интеллектуальным. Но есть большой класс задач, связанных с умением понимать, анализировать, рассуждать, использовать знания, планировать свои действия и т. д., в которых естественно проявление интеллекта (хотя, может быть, не слишком глубокого). Характерной особенностью этих «умений» является их универсальность и поэтому применимость к объектам самой разнообразной природы. Так, умение планировать свои действия подразумевает возможность реализовать эффективное планирование и дома, и на улице, и на работе. А умение оперировать своими знаниями применимо для решения любой практической задачи. Именно поэтому под мето-

дами искусственного интеллекта часто понимают универсальные процедуры (их часто называют метапроцедурами), применимые к широкому классу задач. Что же это за задачи?

Это, прежде всего, задачи, связанные со *зрительным восприятием* компьютера. Более 90 % информации об окружающем мире человек получает именно по этому каналу. Для современного компьютера этот канал практически закрыт, через клавиатуру нельзя пропустить мало-мальский значительный поток информации. Так очень остро встает проблема «открыть глаза» компьютеру. Но ввести изображение в память компьютера нетрудно. Для компьютера сложно понять это изображение.

Понимание изображения — следующий шаг в решении задач *распознавания образов*. Человек, как правило, без труда узнает лица, предметы и знаки (любые буквы, цифры, картинки и т. д.) после однократного знакомства с ними. Для компьютера решение такого рода задач почти всегда затруднительно, а иногда и вовсе пока невозможно.

Другая важная практическая задача — *понимание естественного языка,* текста или речи. Для человека понимание его языка попросту не является задачей вообще. А для компьютера это очень сложная проблема, решить которую в полной мере едва ли удастся в ближайшее время.

Задача *выявления закономерностей* состоит в том, чтобы найти общую закономерность, свойственную всем или большинству рассматриваемых предметов, ситуаций, явлений... Интеллектуальность ее не вызывает сомнений, как и важность для решения сложных задач.

Задача *планирования действий,* например робота, заключается в том, чтобы состав и последовательность этих действий приводили бы к требуемому результату. Например взять определенную деталь и положить ее в другое место заданным образом. Планирование такого рода действий человеку попросту не нужно: он просто выполняет заданную цель (ставит деталь на нужное место и все). Но для робота такое планирование совершенно необходимо: в процессе выполнения задания он может, например, разрушить своим манипулятором то, что было сделано ранее. Планирование действий робота должно осуществляться в его электронном «мозгу» — компьютере. И чем сложнее его действия, тем сложнее задача их планирования и тем мощнее должен быть компьютер, обслуживающий этот робот.

Следующий класс задач искусственного интеллекта связан с понятием *знание.* Мы редко различаем понятия «информация» и «знания». Для нас, людей, знания — нечто более существенное, чем просто информация, и только. Для компьютера разница между знанием и информацией очень большая, поскольку оперировать знаниями для него непривычно. Ведь компьютер не более чем автомат для переработки информации, и прежде чем эффек-

тивно манипулировать знаниями, ему нужно решить много задач. Это задачи представления знаний в памяти компьютера, получения знаний как процесса выявления закономерностей, обучения как способа передачи знания от человека к машине, моделирования рассуждений как задача манипулирования знаниями. Все это задачи искусственного интеллекта (некоторые из них мы рассмотрим в следующих главах).

Как легко заметить, далеко не все из этих задач с человеческой точки зрения следует считать интеллектуальными, ведь две из них — узнавание и планирование действий — без труда решаются даже животными, а понимание речи свойственно самым интеллектуальным представителям рода человеческого. Что же касается знаний и манипулирования ими, то человек всегда ценит знания, но никогда особенно не задумывается о том, как он ими манипулирует. Так что об интеллектуальности большинства приведенных задач и говорить не стоит. Правильнее было бы говорить о машинном (компьютерном) интеллекте, которому предстоит обрести такую «интеллектуальность». Но тем не менее слова «искусственный интеллект» вошли в научно-технический обиход, и мы, следуя традиции, будем так называть эту очень важную область науки и техники, связанную с интеллектуализацией современного компьютера.

А теперь рассмотрим некоторые из этих задач подробнее и покажем, как эти методы искусственного интеллекта используются в компьютерах пятого поколения.

ВОСПРИЯТИЕ ИЗОБРАЖЕНИЙ

Задача компьютерного восприятия изображений опирается на задачу распознавания. Рассмотрим ее применительно к распознаванию графических изображений (в действительности эта проблема значительно шире и затрагивает, например, задачи медицинской и технической диагностики, поиска месторождений, прогноза погоды и многие другие). Трудность решения задачи распознавания графических изображений можно проиллюстрировать на очень простом примере распознавания букв, написанных от руки. Чтобы компьютер мог различать буквы, он должен располагать программой такого распознавания. Эта программа реализует алгоритм, в котором указано, как следует обрабатывать изображение знака, чтобы выяснить, какому образу он принадлежит. Здесь образов столько, сколько букв предстоит распознать. Если бы все изображения одной и той же буквы были бы одинаковы, то проблемы не было бы — всегда легко выявить один-два признака, которые отличают каждую букву от любой другой. Например, острый верх и поперечная черта отличают букву А от Л. Именно такой подход использован в читающих автоматах, сортирующих письма на почте. Наличие шаблонов заставляет нас

писать цифры одинаково, что значительно облегчает задачу читающего автомата. Но как быть с бандеролями и посылками? Их по-прежнему сортируют вручную, так как нет надежного способа (алгоритма) распознавания почерка, т. е. знаков, написанных от руки.

В подобном алгоритме нуждается не только почта. Он нужен прежде всего компьютеру для организации своих действий на основе любой и любым образом написанной текстовой информации, введенной в него. Необходимость действия требует прежде всего понимания компьютером полученной им информации. Решением задачи понимания и занимается распознавание образов, где образом является значение знака, а исходной информацией — его графическое изображение.

Если такого понимания не требуется, то ввести графическую информацию в память компьютера нетрудно — достаточно воспользоваться стандартным телевизионным датчиком и «считать» построчно изображение, чтобы запомнить яркость каждой точки этого изображения. Такая запись, называемая факсимильной, проста, но требуется очень большая память компьютера. А чтобы сократить ее, надо уже понимать смысл этого изображения — распознавать его или отдельные его части. Так, чтобы записать один стандартный черно-белый телевизионный кадр, необходимо запомнить полмиллиона чисел, определяющих яркость каждой точки экрана (для цветного изображения это значение надо утроить). Любопытно, что такой нехитрый способ хранения изображений может быть применен для сохранения разрушающихся картин старых мастеров, точнее, не самих картин, а информации о них в виде поточечной записи на магнитной ленте, примерно так же, как это делается в телевизионной видеозаписи, только с большим разрешением. Известно, что со временем краски изменяются, и нам приходится видеть не то, что видели наши предшественники на этих картинах. Компьютерный способ гарантирует не только сохранность записанной информации, но и передачу ее по каналам вычислительной сети и воспроизведение на экранах с высокой разрешающей способностью. Несложный расчет показывает, что для качественного запоминания небольшой картины нужна память в 1 Гбайт (10^9 байт), т. е. картину можно хранить на одном оптическом диске. Компьютерный способ хранения и передачи изображения уже сейчас широко используется для работы с подписанными документами, чертежами, фотографиями.

Если же изображение содержит элементы, которые можно однозначно распознать, как, например, буквы или цифры, то для запоминания потребуется значительно меньшая память. Так, чтобы запомнить одну страницу текста, требуется всего 1500 байт, каждый из которых соответствует одному из запоминаемых знаков (букв, цифр, знаков пунктуации и т. д.).

Как видно, для сжатия информации, содержащейся в факсимильном тексте, нужно распознать все знаки и записать его так, как записывается в памяти компьютера текст, введенный с клавиатуры, т. е. очень компактно —по одному байту памяти на знак.

Нетрудно представить, как будет использоваться умение компьютера распознавать знаки, например, в радиолюбительской практике. Очень часто радиолюбителю (и не только ему) необходимо знать свойства спроектированной им схемы, чертеж которой имеется лишь на бумаге. Для этого достаточно ввести эту схему в память компьютера, например нарисовав ее световым пером на экране дисплея. В памяти компьютера окажется факсимильное изображение этой схемы со всеми необходимыми знаками и надписями, сделанными от руки «коряво». Программа должна распознать элементы этой схемы, ее структуру и надписи, после чего схема считается «понятой» компьютером. Дальнейшая обработка — решение соответствующих уравнений и представление результата радиолюбителю — происходит «старым», уже известным образом. Здесь основная трудность заключается в процессе распознавания «корявой» схемы, введенной в память компьютера. Пятое поколение машин должно владеть этими методами в полной мере.

ВЫЯВЛЕНИЕ ЗАКОНОМЕРНОСТЕЙ

Эта проблема обобщает задачи зрительного восприятия вообще и распознавания образов в частности. Действительно, ведь образ — не что иное, как простейшая закономерность, выявленная на множестве изображений, представленных для распознавания. Например, чтобы одни изображения назвать буквой А, а другие — буквой Б, нужно выявить закономерность в написании этих букв, но подобные закономерности имеются не только в этих изображениях, но и в числовых таблицах, текстах, процессах и т. д.

Примером задачи выявления закономерности служит известная игра «Кто лишний?». Заключается она в следующем. Предъявляется несколько объектов и из них предлагается исключить лишний, причем признаки его не указываются — их нужно найти самому, т. е. вывести закономерность, связывающую все объекты, кроме одного, который и будет лишним. Например, из четырех животных: кошка, собака, курица, медведь — выделить лишнего. Не торопитесь с ответом — их может быть много! Действительно, это может быть и курица (единственная птица), и кошка (только она способна втягивать когти), и собака (только она обладает «собачьим» нюхом), и медведь (самый большой). Конечно же, первый ответ наиболее распространен: деление животных на зверей и птиц важно для человека. Но в определенных обстоятельствах могут быть выделены и другие закономерности. Как видно, процесс выявления

закономерностей сильно подвержен внешнему влиянию и существенно зависит от предварительной (априорной) информации, например от цели выявления этой закономерности.

Для чего нужно выявлять закономерности и как этой возможностью воспользуются компьютеры пятого поколения? Прежде всего отметим, что процесс выявления закономерностей не что иное, как важнейший элемент процесса познания окружающего мира. Действительно, мы считаем, что знаем объект, если имеем представление о закономерностях его поведения, можем предсказать, как он поведет себя в той или иной ситуации. Здесь под поведением понимаются свойства, признаки, характеристики, структура и т. д.— все то, что отличает его от других объектов и важно для нас. Знания такого рода очень ценны при общении с этим объектом. Они могут быть использованы для управления, приспособления этого объекта к нашим нуждам.

Это свойство выявления закономерностей очень нужно роботам, которым предстоит действовать в новой ситуации (например, на дне моря, в жерле вулкана, на другой планете и т. д.). Прежде чем действовать, роботу нужно иметь представление об объектах, его окружающих, выявить закономерности их поведения. Программа выявления закономерностей и будет реализовывать процесс познания этого робота. Результатом будут закономерности (или модели) среды, в которой действует робот. Эти модели и будут использованы при организации его действий, направленных на достижение поставленных целей.

Другой важной технической задачей выявления закономерностей является сжатие информации. Дело в том, что современные средства сбора информации настолько производительны, что очень быстро «забивают» практически любую компьютерную память. Например, стандартный телевизионный датчик требует размещения 10 млн. байт информации каждую секунду. Легко вычислить, что он очень быстро перегрузит любую память. Именно поэтому надо сжать такой обильный поток информации. Для этого достаточно выявить его закономерность и фиксировать лишь отклонения от нее, которых, естественно, не будет много (при правильном выявлении закономерностей, разумеется).

Как же решается задача выявления закономерностей? Ее основой является индукция — способ суждения от частного к общему. Напомним, что индукция в определенном смысле противоположна дедукции — суждению от общего к частному по схеме: все объекты типа A обладают свойством B, конкретный объект C принадлежит к типу A, следовательно, он имеет свойство B. Например, все рыбы (это тип A) плавают (это свойство B), следовательно, карась (это объект C) плавает (т. е. обладает свойством B). (Здесь приведено лишь очень узкое толкование понятия дедукции, которое в широком смысле связано

с выводом следствия из заданных утверждений (посылок). Именно это определяет дедукцию как основу построения любого теоретического знания, где такой вывод играет определяющую роль. В частном (аристотелевском) смысле дедукция является механизмом, позволяющим переходить от общего суждения к частному.)

В индукции же наоборот. По частным конкретным проявлениям (реализациям) следует вывести общую закономерность. Действует индукция по схеме: если некоторые объекты A_1, A_2, ..., A_k принадлежат к типу A и обладают свойством B, то все объекты типа A имеют свойство B. Например, индуктивным является суждение, что все рыбы плавают по наблюдению за тем, что плавает карась. Индукция — основной и очень эффективный инструмент эмпирического знания. Именно индукция позволяет превращать его в теоретическое знание, т. е. в общие суждения. Но в отличие от дедукции, чтобы сделать правильным индуктивный вывод, необходимо иметь еще некоторые представления об объекте индукции, позволяющие не ошибиться. А ведь ошибиться в процессе индуктивного вывода так легко!

Отметим рискованность всякой индукции. Действительно, по отдельным частным наблюдениям необходимо вывести общую закономерность, которая должна соответствовать всем остальным наблюдениям, которые неизвестны. Здесь очень легко ошибиться. Так, зная, что карась, карп и селедка покрыты чешуей, можно заключить по индукции, что все рыбы имеют чешую. Однако эта закономерность ошибочна: есть рыбы без чешуи (например, сом, акула). Именно поэтому так трудно реализовать эффективный механизм индукции и заключение о выявленной закономерности всегда сопровождают осторожными оговорками «часто», «скорее всего», «вероятно» и др.

Примером реализации простейшего механизма индукции является использование так называемого случайного поиска. Он сводится к случайному изменению исходной закономерности и проверке эффективности этого изменения на имеющихся наблюдениях. Если закономерность в результате случайного изменения стала хуже описывать наблюдения, то следует вернуться к исходной и снова изменить ее случайно. Если же она стала лучше, то следует ее считать исходной и вводить следующее случайное изменение и т. д. Теория и практика подтверждают, что такой способ выявления закономерностей всегда приводит к результату, хотя иногда и требует много времени.

Закономерность является одной из форм знания, и поэтому так важно компьютерам пятого поколения уметь выявлять закономерности, чтобы пополнять свои знания. (О том, как представляются знания в компьютере, мы расскажем в следующей главе.)

ПОНИМАНИЕ ЕСТЕСТВЕННОГО ЯЗЫКА

Эта проблема одна из самых трудных в искусственном интеллекте. Ее сложность мы проиллюстрировали в гл 9 на примере задачи перевода с одного естественного языка на другой. Лет тридцать назад думали, что проблема машинного перевода несложна и скоро будет решена. Предполагали, что в 70-х годах компьютер будет бойко переводить любой текст с английского на русский и наоборот. Этот прогноз до сих пор не оправдался. Считалось, что для эффективного перевода достаточно ввести в память компьютера англо-русский словарь и известные правила грамматики обоих языков. Но из этого ничего не получилось. И виной тому стала, прежде всего, многозначность естественного языка — одни и те же слова имеют разный смысл в зависимости от контекста и знаний об окружающем мире. Если компьютеру, не владеющему необходимыми знаниями, сказать: «Здрасте! Я — ваша тетя», то он будет считать, что с ним здоровается его тетка, он не понял в этих словах иронии, которую высказывают по поводу явно нелепого результата.

Такого рода знания уже выходят за рамки знаний о языке. Это знания о мире, в котором живет пользователь. Для правильного понимания пользователя компьютер должен знать о мире то же, что знает пользователь. Эти знания сосредоточены в *базе знаний*, которую должен иметь всякий компьютер, претендующий на понимание естественного языка, чтобы правильно истолковывать каждое полученное сообщение (подробнее с базами знаний мы познакомимся в гл. 12). Компьютерная база знаний всегда ориентирована на определенную предметную область, в рамках которой работает пользователь. И делается это вовсе не из-за лености разработчиков баз знаний. Просто объем базы знаний на все случаи жизни слишком велик, ведь она должна вмещать все знания, накопленные человечеством! Проблема здесь даже не в том, как разместить такой объем информации в памяти компьютера, а в том, как получить эту информацию. Так что общение с компьютером пока не может быть на любую тему — создать такую базу знаний пока еще (а может быть, и вообще) невозможно. Да и понадобится ли она в дальнейшем? Едва ли стоит тратить столь много усилий, чтобы просто поболтать с компьютером. Так что и с компьютером пятого поколения придется общаться на проблемно-ориентированном естественном языке.

А как же бытовые темы? Ведь компьютерам пятого поколения предстоит войти в каждый дом и общаться с любым человеком. Здесь следует иметь в виду, что запросы пользователя о товарах, расписании поездов или сеансах кино и т. д. являются проблемно-ориентированными разговорами. Но «задушевного»

общения с компьютером пятого поколения не будет. Это дело компьютеров будущих поколений.

ПЛАНИРОВАНИЕ ДЕЙСТВИЙ

Эта проблема значительно шире, чем мы ее представили в начале главы. Действительно, планировать свои действия приходится не только роботу, получившему задание. Планировать необходимо при решении любой сложной задачи. Актуальность автоматизации решения задачи планирования действий с каждым годом возрастает, все больше сложных задач нуждается в решении на компьютере. Это задачи робототехники, научно-исследовательские, задачи проектирования, народнохозяйственного планирования, управления и многие-многие другие.

Структура такого рода задач довольно проста — они держатся на трех «китах»: исходная ситуация, целевая ситуация и модель среды, в которой приходится планировать действие. Все эти три компоненты необходимы при решении задачи планирования. Например, при планировании действий робота, кроме исходного состояния, необходимо знать целевую ситуацию — какой должна быть среда в результате действий робота. Моделью среды здесь являются его знания относительно того, что произойдет, если будет выполнено то или иное действие. Располагая этой исходной информацией, робот (а точнее, компьютер, являющийся его «мозгом») планирует свои действия так, чтобы исходную ситуацию свести к целевой. Для этого ему нужно уметь оценивать близость сложившейся ситуации к целевой. Очевидно, что планирование действий сводится к тому, чтобы складывающаяся в результате этих действий ситуация была бы как можно ближе к целевой (при соблюдении всех ограничений, разумеется). Действуя так, компьютер «добирается» до целевой ситуации. Так и решается задача планирования действий. Можно действовать и наоборот, двигаясь от целевой ситуации к исходной, что часто бывает удобней и проще.

Здесь, однако, следует отметить, что далеко не всегда критерий локального успеха приводит к цели. В жизни, к сожалению, мы очень часто встречаемся с ситуациями, когда «окольной дорогой ближе, чем прямой», когда надо трезво оценивать возможности. Именно такая оценка заставляет при планировании действий часто отказываться от успеха на каждом шагу, а иногда удаляться от цели, но так, чтобы быстрее достичь ее на последующих шагах.

Так или иначе, но план действий строится на модели среды. Процесс этот сложен по трем причинам. Во-первых, модель среды как описание состояния ее элементов и их взаимодействия всегда получается очень громоздкой. Кроме этого, модель среды должна сопровождаться знаниями о законах, действующих

в ней, например нельзя предметы ставить на острые грани, и, наконец, результат какого-то действия нельзя предвидеть заранее — нужно сначала «разыграть» его на модели. Это придает процессу планирования характер проб и ошибок, на что, естественно, приходится затрачивать значительные вычислительные (точнее, моделирующие) ресурсы компьютера. Следовательно, с задачей планирования действия в сложных ситуациях смогут справиться лишь компьютеры пятого поколения, особенно для робота, действующего в быстро изменяющихся ситуациях, когда он должен принимать оперативные решения.

В заключение следует отметить, что методы искусственного интеллекта, которые станут основными методами решения задач компьютерами пятого поколения, обладают следующими важными свойствами.

1. Это технология решения различных проблем в различных областях применения (а не продукция), именно поэтому методы искусственного интеллекта часто называют новой информационной технологией.

2. Широкое применение этой технологии не может не изменить аппаратные средства и, безусловно, окажет влияние на архитектуру компьютеров пятого поколения. Это значит, что методы искусственного интеллекта не только будут программной «начинкой» компьютера, но и определят его структуру, конструкцию его блоков.

3. Именно искусственный интеллект открывает новую эру интеллектуальных машин, с которыми будет общаться не только человечество, но и каждый его представитель. И самым решительным шагом в этом направлении будет создание экспертных систем, в которых методы искусственного интеллекта найдут, пожалуй, наиболее эффективное (и эффектное) применение. Это позволит компьютеру в ряде областей стать умнее (без кавычек) человека-специалиста. (Об этом будет рассказано в гл. 14.)

— Когда речь идет о дедукции,— заметил Мегрэ,— я не могу не вспомнить моего знаменитого коллегу Шерлока Холмса с его дедуктивным методом. Он так ловко раскрывал преступления, пользуясь этим методом, что заинтересовал меня профессионально. Я, конечно, понимаю, что описание механизма раскрытия преступления в художественной литературе далеко от реальности. Но, возможно, что-то и можно использовать в нашей повседневной практике. Ведь дедуктивный метод, придуманный Конан Дойлем, наверное, поддается формализации, может быть запрограммирован и введен в память компьютера. Нельзя ли такой компьютер приспособить для криминального розыска? Что вы думаете по этому поводу, Поль?

— Вы правы, дедуктивный метод легко формализуется,— грустно ответил Поль,— но метод, описанный Конан Дойлем, никак нельзя считать дедуктивным. К сожалению, эту ошибку, давно замеченную специалистами-логиками, никак не могут понять криминалисты. По-видимому, обаяние личности Шерлока Холмса так

велико, а впечатление от книг, прочитанных в юности, так сильно, что именно криминалисты продолжают настаивать на дедуктивности метода Конан Дойля.

— А что же это такое? — изумился Мегрэ.

— Это типичный индуктивный метод. И это легко доказать. Помните, как великий сыщик по мельчайшим деталям воссоздал картину преступления? Такой способ и есть индукция. Дедукция же дает возможность заключений лишь от общего к частному, например, искать того, кому это выгодно. Здесь общим положением является утверждение, что всякое преступление не случайно, а результат злой воли, которой оно было необходимо или выгодно. Поэтому всякий, кого устраивают последствия преступления, может подозреваться в соучастии. Таков дедуктивный вывод.

— Ну знаете, чтобы сделать такой «вывод», ума не надо. Это общеизвестная истина. Неужели настоящий дедуктивный метод дает столь тривиальные выводы? — спросил Мегрэ.— Если «да», то грош (точнее, сантим) ему цена.

— Разумеется, нет,— поспешно ответил Поль.— Дедуктивный метод — это метод теоретического знания, которое само по себе является обобщением, ведь всякая теория — это обобщение частных фактов. Располагая теорией, мы всегда можем выводить ее частные следствия, порождать новые факты, точнее, не факты, а положения, которые могут стать фактами при сопоставлении их с действительностью. И именно это обеспечивает дедукция — из общих соображений (теория) выводить конкретные факты (следствия). Это очень сильный инструмент, но для его применения требуются весьма большие и хорошо организованные знания.

— А если все-таки попробовать применить дедуктивный метод в криминалистике. Что из этого вышло бы?— полюбопытствовал Мегрэ.

— Для этого нужно иметь общую теорию преступления, такую его модель, из которой как частные случаи следуют все возможные конкретные преступления. Располагая такой моделью и «подставляя» в нее конкретные факты, полученные на месте преступления, можно «вывести» всю картину преступления, ведь это лишь частный случай! Не правда ли, было бы очень заманчиво?

— Да,— улыбнулся Мегрэ,— дело за малым, за такой моделью. Как только она будет сделана, я ухожу на пенсию: человеку делать будет нечего. Расследование будет вести компьютер! К счастью, этого не произойдет, создать такую модель нельзя. В ближайшее время, разумеется,— добавил он ехидно.

— Но бог (или черт) с ней, дедукцией. Поговорим об индукции в нашем деле. Она не требует теории, как дедукция. И это мне нравится. А чего она требует?

— Комиссар, когда вы почините ваши каминные часы?

— А вы откуда знаете, что они испортились, ведь они остановились только сегодня ночью! — изумился Мегрэ.

— Я определил это по кофейному пятну на вашем галстуке.

— А какая связь между пятном на моем галстуке и ходом каминных часов? — спросил Мегрэ.— Сержант Поль! Прекратите разыгрывать меня. Вы были сегодня в моем доме? И видели остановившиеся часы?

— Нет, шеф,— улыбнулся Поль.— Я вывел этот факт индуктивно. Зная вашу

аккуратность, я понял, что вы торопились и поэтому не успели сменить галстук. А зная вашу пунктуальность, я понял, что ваша торопливость была вызвана объективными причинами. А так как вы были совершенно спокойны весь день, я понял, что мадам Мегрэ здорова и никаких неприятностей у вас не было. Оставалась одна простейшая причина — вы просто проспали, так как часы остановились и не пробили вовремя.

— Да, но могло быть все совсем по-другому!

— Конечно! И в этом рискованность всякой индукции. Но, кроме риска, у нее есть еще одно важное свойство. Если дедуктивный вывод может сделать любой (при наличии теории, разумеется), то для того, чтобы сделать индуктивный вывод, нужно многое знать по поводу объекта вывода и суметь это использовать. Я должен был многое знать, чтобы заключить, что ваши часы остановились. Любой ваш случайный посетитель, наверное, решил бы, что вы просто небрежны.

— Вы утверждаете, что я действую по индуктивному методу? Стало быть, я кое-что знаю и умею.

— Конечно! Простак, даже самый исполнительный, не может быть хорошим сыщиком. Помните капитана Лестрейда из Скотланд Ярда. Этот служака не был обременен знаниями и именно поэтому на каждом шагу попадал впросак.

— Но неужели Конан Дойль так легко и грубо ошибся, назвав индуктивный метод дедуктивным? Ведь логика — очень старая наука, и понятия индукции и дедукции сформировались еще во времена Аристотеля.

— Думаю, что Конан Дойль обратил внимание лишь на одну сторону расследования — дедуктивную. Наверное, он под этим понимал не собственно дедукцию, а логический метод вывода, который опирается и на дедукцию, и на индукцию.

— Да, пожалуй, нам приходится пользоваться и тем, и другим. Когда по имеющимся на месте преступления следам и фактам надо восстановить картину преступления, построить версию, то здесь мы занимаемся индукцией — от частного к общей картине. А когда надо проверить справедливость версии или отвергнуть ее, то действуем дедуктивно — проверяем справедливость частных следствий из этой версии. Все следствия версии должны подтверждаться. Только тогда версия доказана и становится поводом для обвинения (или оправдания). Если же хоть одно следствие из версии не подтверждается, то эту версию следует отвергать или корректировать. Эта самая трудная часть. Трудная не только потому, что надо создавать новую версию, согласующуюся со всеми имеющимися фактами, но и потому, что сам поверил в эту версию (иначе не выдвигал бы ее). И так хочется отвергнуть не версию, а факты, противоречащие ей, или поставить их под сомнение.

— Вот именно для этого и нужен компьютер,— весело заметил Поль.— Ведь он не знает эмоций и пристрастий. И будет генерироваь и проверять новые версии до тех пор, пока не будет найдена неопровержимая.

— Что ж, может быть, доживем и до этого.

12. ТЯЖКИЙ ПУТЬ ПОЗНАНИЯ

(Представление знаний в компьютере)

ЧТО ТАКОЕ ЗНАНИЕ?

Четкий ответ на этот вопрос дан в философском словаре. Знание — это «идеальное выражение в знаковой форме объективных свойств и связей мира, природного и человеческого». Против такого общего определения трудно что-либо возразить, но и использовать его также трудно. Конкретизируем его. Назовем знанием набор моделей об окружающем нас мире. Такого рода знания являются прямым следствием потребностей человека эффективно действовать в этом мире. А чтобы действовать, надо прежде всего иметь представление о том, с чем будешь иметь дело, о тех предметах, явлениях, процессах — вообще объектах, с которыми придется взаимодействовать. Чем лучше эти представления, тем, естественно, успешнее будет действие, связанное с этими объектами. Именно поэтому так важны правильные представления, знания об окружающем мире. Это и есть модели.

Знания удобно подразделять на стратегические, которые могут оказаться полезными в будущем, и тактические, отражающие специфику сложившейся ситуации и нужные для образования сиюминутного поведения. И тот и другой вид знания совершенно необходим для полноценной деятельности. Действительно, отсутствие стратегического знания делает человека беззащитным в критических ситуациях, когда требуется быстро принять эффективное решение и нет времени. Без тактического знания, учитывающего особенность и специфику сложившейся ситуации, человек попросту лишен возможности приспосабливаться к окружающей среде. Если без тактического знания вообще нельзя обойтись, то без стратегического — можно, но очень плохо. Чем больше запас стратегического знания, тем эффективнее можно строить свое поведение. Запасание этих знаний идет впрок и без гарантии, что они понадобятся в дальнейшем (ведь нельзя представить всех ситуаций, в которые попадет индивидуум в будущем). Так, львиная доля знаний, получаемых нами в детстве, стратегического характера и поэтому зачастую не используется во взрослой жизни.

Такая двойственность стратегических знаний — необходимость в критических ситуациях и явная избыточность — неизбежная плата за их универсальность. И именно стратегические знания, их эффективное представление в памяти компьютера, чтобы оперативно использовать их для решения поставленных задач, и является предметом забот при создании интеллектуальных компьютерных систем и одной из злободневных проблем компьютерного интеллекта.

Сделаем следующий шаг в конкретизации понятия знания.

И этот шаг приводит к модели. Это вовсе не означает, что знание и модель эквивалентны. Знание шире модели, но всякая модель является, безусловно, знанием. Более того, знания в основном состоят из моделей. Это и дает нам основание рассматривать знание как модель.

Модель — очень удобная форма знания об окружающем нас мире. Всякое знание связано с объектом, относительно которого это знание имеет место. Например, кадровая анкета представляет собой знание о человеке. Но знание это неполное, усеченное, куцее, и тем не менее оно в некотором, очень узком, смысле является эквивалентом человеку (как это ни звучит парадоксально). Более того, эта анкета часто заменяет человека при решении некоторых кадровых вопросов.

Идея замены объекта некоторым его простым информационным эквивалентом и лежит в основе процесса познания. Таким эквивалентом является модель объекта. Познавая окружающий мир, мы строим модели явлений, процессов, предметов и т. п.— объектов этого мира.

Итак, модель — это информационный эквивалент объекта, созданный для достижения определенных целей. Очевидно, что с изменением целей меняется и модель объекта. Например, кадровая анкета молодого работника существенно отличается от личной «анкеты», которую ведут его молодые сотрудники по работе. Оно и понятно: цели составления этих анкет очень разные.

Но было бы неправильно ограничивать модели только анкетой объекта. Анкета — лишь простейшая форма модели, она содержит перечень некоторых свойств объекта и может заменить его лишь в том случае, если необходимо знать именно эти свойства. А если понадобились другие? Ведь принимая решение относительно какого-то объекта мы прежде всего интересуемся свойствами, нужными для этого решения, а не теми, которыми располагаем. Вот и получается, что анкета, как бы подробна она ни была, часто не удовлетворяет потребностям. Волей-неволей приходится обращаться к самому объекту. Именно этим свойством анкеты как модели пренебрегают бюрократы, когда принимают решения относительно работника по его анкете, за что их давно и заслуженно критикуют.

Более полная модель объекта должна содержать не только сведения о его измеренных свойствах, но и о тех, которые не измерялись. Возможно ли это? Возможно, но, конечно, в ограниченной мере. Например, зная, что в заданном режиме объект имеет определенное свойство, естественно считать, что при небольшом изменении этого режима указанное свойство изменится тоже незначительно. Такая экстраполируемость данных позволяет определять на модели свойства объекта, которые не измерялись заранее. Это и есть самое ценное свойство модели — не быть «складом» исходных сведений об объекте, его анкетой, а предо-

ставлять возможность судить о возможном поведении объекта в ситуациях, в которых его свойства не измерялись, т. е. не записаны в анкете. Например, можно по кадровой анкете определить цвет волос человека? Наверное, нельзя. А по медицинской анкете — по истории его болезни? Можно, в какой-то мере, если привлечь необходимые сведения о связи цвета волос с различными заболеваниями (известно, например, что рыжеволосые чаще других болеют аллергическими заболеваниями).

ДАННЫЕ И ЗНАНИЯ

Итак, всякая модель позволяет судить об объекте, который она описывает. Это знания, на основе которых мы можем эффективно использовать этот объект, например управлять им для достижения каких-либо своих целей. Теперь попробуем эти наши знания передать компьютеру, чтобы он с их помощью смог бы так же управлять объектом.

Но прежде чем модель ввести в память машины, мы должны представить ее в виде знаков, чтобы компьютер мог манипулировать этой моделью в процессе управления объектом (ведь модель является эквивалентом объекта и синтез управления осуществляется с помощью этой модели — именно для этого прежде всего и нужна модель). Итак, вводя модель в память компьютера, следует прежде всего помнить, что с ней придется работать. Поэтому модель традиционно стараются представить в виде аналитического выражения (формулы, уравнения и т. п.), так как с такими выражениями легко работать компьютеру. Пользуясь аналитической моделью, мы задачу управления сводим к чисто математической (точнее, вычислительной) задаче определения значения некоторых интересующих нас переменных, которые характеризуют будущее управление объектом. Очевидно, что для решения такой вычислительной задачи не нужно знать, какой именно смысл имеют те или иные параметры и переменные модели. Действительно, нам совершенно безразлично, какой содержательный смысл имеет переменная x в квадратном уравнении $ax^2+bx+c=0$. И лишь получив ее мнимое значение, мы начинаем беспокоиться, да и то только в том случае, если по мысли задачи x — реальная величина.

Именно такого рода сведения, смысл которых не существен для манипуляции с ними, называют *данными*. Как видно, данные являются результатом «расщепления» знания на смысл (хранимый человеком) и данные — «бессмысленные» знаки. Например, любой текст, введенный в память компьютера, является только данными и лишь наличие механизма интерпретации (осмысливания), способа использования этих данных, превращает их в знания.

Традиционная схема использования компьютера связана именно с обработкой данных, а носителем и хранителем знаний

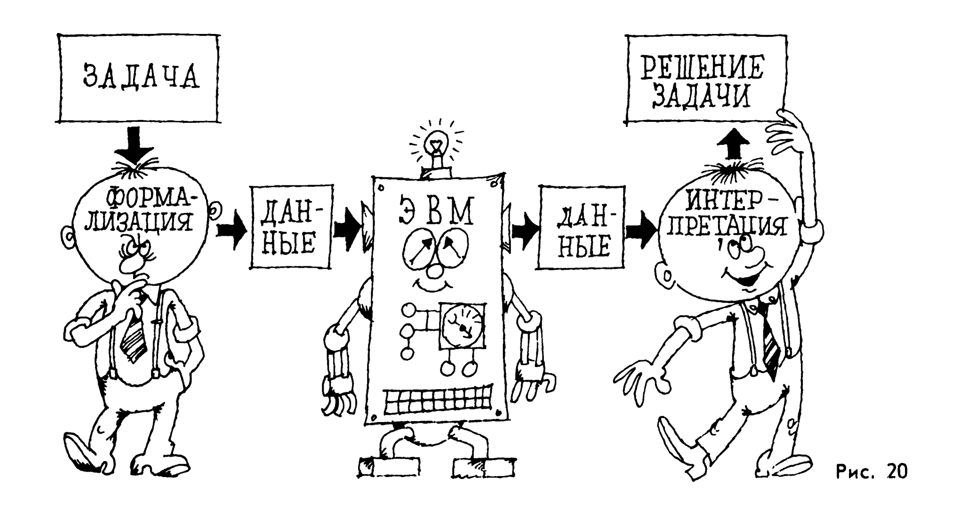


Рис. 20

является пользователь. на которого и возлагается функция «отщепления» данных от знания (это стадия формализации) и интерпретация полученных результатов, т. е. осмысливание (рис. 20). Как видно из рисунка, результатом являются тоже только данные, а не решение задачи, которое получается только после интерпретации их пользователем.

Описанная схема решения задачи с помощью компьютера не является чем-то необычным. Если в ней заменить слово «компьютер» на «решатель», в качестве которого может выступать человек, то получим обычную схему решения задач с привлечением формальных методов (не обязательно компьютерных). Это путь развития современной науки начиная с Галилея. Здесь формализация и интерпретация выступают в роли своеобразных антиподов: формализация как «отщепление» данных от знания и интерпретация как внесение смысла в данные, образование знания.

Можно предложить довольно грубую «формулу» знания:

Знание = данные + их смысл

Именно такое представление знания стимулировало развитие формальных методов обработки информации в общем и вычислительных методов в частности. Действительно, умение решать математические задачи позволяет решать все задачи, которые сводятся к математическим. Именно это привело к возрастанию роли и престижа формальных методов в науке, технике и народном хозяйстве. Формализация стала тем универсальным средством, с помощью которого решались все задачи, поддающиеся формализации. Хорошая постановка задачи в науке соответствует ее формализации, дальнейшее ее решение «дело техники».

Такая методология привела к очень интенсивному росту знаний, который наблюдался в науке нового времени и начало которому положил Галилей. Смысл этой методологии коротко сводится к следующему. Всякий «формализм» работает только с данными, а носителем смысла является только человек. Это положение было незыблемо более трехсот лет. Но вот настало время усомниться в нем.

ТРУДНОСТИ ФОРМАЛИЗАЦИИ

Они видны хотя бы из экспериментов по машинному переводу с одного языка на другой (об этой проблеме мы подробно говорили в гл. 9). Результаты этих экспериментов были очень несовершенны, но тем не менее они показали, что компьютер может оперировать со смыслом предложения в процессе перевода с одного языка на другой, может понимать текст. А это означает, что компьютер может манипулировать не только данными, но и знаниями, подобно человеку.

Это обстоятельство важно не только с познавательной точки зрения (машина может работать со знаниями), но прежде всего для практических применений. Дело в том, что процесс формализации ограничен тем математическим аппаратом, который имеется в распоряжении человечества. Это мощный аппарат, но, увы, ограниченный. Далеко не всегда и не все задачи удается формализовать. Что же это за задачи? Прежде всего, это задачи организационного управления, т. е. управления системами с людьми. Говорят, что они имеют дополнительные степени свободы, связанные с поведением человека. И именно они заставили поставить задачу введения знаний (а не только данных) в компьютер. Так возникла проблема представления знаний в ЭВМ.

Рассмотрим способы решения этой проблемы. Первой и очень естественной мыслью является обращение к естественному языку. Ведь это самое совершенное средство описания любых объектов. Мы, люди, великолепно используем его для хранения и передачи знаний. Нет предмета, вещи, явления, процесса, которые нельзя было бы описать на естественном языке. Это описание предшествует всякой формализации и заканчивает ее на стадии интерпретации, которая происходит тоже на естественном языке.

Всем хорош естественный язык. Но ряд недостатков, точнее досадных свойств (о которых мы говорили в гл. 9), препятствует его прямому применению для представления знаний в компьютере. Но, «если гора не идет к Магомету, Магомет идет к горе». Если естественный язык чем-то не подходит, надо построить свой искусственный, максимально похожий на естественный, но без его недостатков. И такой язык был создан, и не один. Рассмотрим,

например, язык, который был сконструирован специально для решения задач управления сложными объектами. В процессе управления всякими сложными объектами необходимо строго контролировать сложившуюся ситуацию, описывать ее на удобном языке, позволяющем анализировать эту ситуацию и строить управление. Такое управление назвали ситуационным, а язык, позволяющий описывать сложные ситуации, ...

ЯЗЫК СИТУАЦИОННОГО УПРАВЛЕНИЯ

Если несколько огрубить наши представления об окружающем мире, то их можно свести к двум основным категориям: понятиям и отношениям между этими понятиями. Действительно, все существующие предметы, вещи, явления и их свойства (все это и есть понятия) связаны между собой какими-то отношениями (находиться в, иметь цвет, передвигаться к, быть одновременно с и т. д.). Например, факт того, что чашка находится на столе, фиксирует отношение «находится на» между двумя понятиями — чашкой и столом. Это отношение или имеет место, если действительно находится на столе, или нет, если эта чашка стоит, например, на стуле, а на столе ее нет. Таким образом, для двух заданных понятий заданное отношение или имеет место, или нет. Третьего не дано, дробного отношения не существует. Именно поэтому отношения имеют бинарный характер, а язык, использующий такие отношения, называют также языком бинарных отношений. Заметим сразу, что эта бинарность также огрубляет наши представления об окружающем мире. Такова судьба всех моделей — для их простоты приходится упрощать реальное явление. Без этого познание было бы невозможно, так как всякое реальное явление бесконечно сложно, его нельзя охватить полностью в конечной модели, да и не нужно.

В чем огрубление бинарности отношений? Да в том, что некоторые отношения только с большой натяжкой можно назвать бинарными. Например, отношение «любить» наверняка имеет множество градаций, хотя некоторые и настаивают на его бинарности, задавая извечный вопрос: «Ты меня любишь?»

Исключим подобные случаи и будем рассматривать лишь бинарные отношения, которые для описания технических знаний действительно можно без натяжки считать бинарными. Теперь опишем язык ситуационного управления как язык бинарных отношений, с помощью которого очень легко описывать факты, ситуации и тому подобные категории, которые входят в то, что мы называем знанием.

Язык этот очень прост. Как всякий язык, язык бинарных отношений имеет свой словарь и свою грамматику. Рассмотрим их в отдельности.

Словарь

Состоит из трех разделов, содержащих слова трех типов.

Прежде всего это *понятия*; будем обозначать их a_i, $i=1$, ... Понятия порождаются описываемым явлением. Проще говоря, это элементы, из которых состоит описываемый объект, так как нас интересуют знания об этом объекте, например a_1 — деталь, a_2 — конвейер, a_3 — оператор и т. д.

Имена, т. е. конкретные значения имен элементов, входящих в описываемую ситуацию; будем обозначать их b_j, $j=1$, ... Например, b_1 — № 268, b_2 — Иванов, b_3 — Петров.

Отношения; будем обозначать их r_k, $k=0$, 1, ... Отношение r_0 строго специализировано для введения имен: r_0 — «иметь имя». Другие отношения вводятся в зависимости от специфики описываемого объекта, например r_1 — «находиться на», r_2 — «управлять», r_3 — «быть одновременно с» и т. д.

Таков словарь. Он строится в зависимости от особенностей описываемого объекта или ситуации. В лингвистике есть понятие «подъязыков». Это языки, порожденные определенной узкой предметной областью, например физикой, химией, математикой, международными отношениями. Так вот, словарь ситуационного управления является словарем подъязыка управляемого объекта, который описывается этим языком, и предметной области, которой принадлежит этот объект. Например, словарь ситуации «Деталь № 268 находится на конвейере, которым управляет оператор Иванов»:

$$\{a_1, a_2, a_3, b_1, b_2, r_0, r_1, r_2, r_3\},$$

где (повторим сказанное выше для удобства в дальнейшем)

a_1 — деталь, a_2 — конвейер, a_3 — оператор — понятия;

b_1 — № 268, b_2 — Иванов — имена;

r_0 — иметь имя, r_1 — находиться на, r_2 — управлять, r_3 — быть одновременно с — отношения.

Сразу можно заметить, что в словаре нет слова «который», так как оно лишнее. Эту ситуацию можно записать более компактно (хотя и не очень стилистически грамотно): «Деталь № 268 находится на конвейере, управляемом оператором Ивановым», но и это не подходит — не ясно, как Иванов управляет конвейером: сейчас или вообще умеет это делать. Здесь появился причастный оборот, свойственный естественному языку (как и придаточные предложения), от которого хотелось бы избавиться. Для этого запишем ситуацию в скобочной форме: (деталь № 268 находится на конвейере) одновременно с (оператор Иванов управляет конвейером). Здесь все понятно и однозначно, хотя и очень «деревянно».

Грамматика

Напомним, что грамматика состоит из морфологии (науки о формах слов) и синтаксиса (науки о соединении слов в предложения). Морфологии в языке ситуационного управления нет: его слова однобуквенные. А синтаксис состоит в том, как определяется понятие «правильная фраза» на этом языке. Имеется в виду синтаксически правильная фраза, семантически (по смыслу) она может быть и ошибочной.

Итак, правильной фразой на языке ситуационного управления является любая фраза, удовлетворяющая одному из трех правил:

1. Любое понятие является правильной фразой.

2. Если A и B — правильные фразы, то (ArB) — тоже правильная фраза, где r — любое отношение, кроме r_0 — «иметь имя».

3. Если A — правильная фраза, то (Ar_0b) — тоже правильная фраза, где b — любое имя.

Вот и все! Проще грамматики придумать нельзя.

А теперь запишем нашу ситуацию на этом языке. Сначала выпишем простейшие правильные фразы:

$A_1=(a_1r_0b_1)$ — деталь № 268 (точнее, «деталь имеет имя № 268»);

$A_2=(a_3r_0b_2)$ — оператор Иванов (точнее, оператор имеет имя Иванов).

Здесь мы воспользовались 3-м и 1-м правилами грамматики языка бинарных отношений. Пойдем дальше и воспользуемся 2-м правилом:

$A_3=(A_1r_1a_2)$ — деталь № 268 (A_1) находится на (r_1) конвейере (a_2);

$A_4=(A_2r_2a_2)$ — оператор Иванов (A_2) управляет (r_2) конвейером (a_2).

Теперь нетрудно описать всю ситуацию, введя одновременно ситуации A_3 и A_4:

$(A_3r_3A_4)$ — деталь № 268 находится на конвейере (A_3), которым в данный момент управляет оператор Иванов (A_4).

Если подставить сюда все имеющиеся выражения, то получим описание ситуации через элементы, образующие исходный словарь:

$$(\underbrace{(\underbrace{(a_1r_0b_1)}_{A_1}r_1a_2)}_{A_3}r_3\underbrace{(\underbrace{(a_3r_0b_2)}_{A_2}r_2a_2)}_{A_4})$$

Вот и все! Но все ли?

Скептически настроенный читатель наверняка скажет, что это лишь замена одних обозначений другими, т. е. слов естественного языка знаками — буквами латинского алфавита с индек-

сами. Действительно, такая замена имеет место, но она не механическая, а с учетом смысла предложения путем сведе́ния всей сложности естественного языка лишь к простой конструкции правильной фразы, связывающей две другие фразы определенным отношением. Причем таких отношений оказывается не слишком много. Так, для русского языка их выделено около 180. И никаких падежей, склонений, спряжений и всякого рода оборотов (причастных и деепричастных) вместе со сложноподчиненными (придаточными) и сложносочиненными предложениями и т. п. сложностями грамматики естественного языка. Вместо изобилия правил и исключений естественного языка — лишь три простых правила без исключений. Безусловно, это значительное упрощение, сохраняющее смысл исходного текста на естественном языке.

Очевидно, что преобразование текста из естественного языка в язык ситуационного управления нельзя делать формально, нужно сохранить смысл, используя имеющийся словарь и наращивая его новыми словами при необходимости.

Одной из наиболее существенных черт этого языка является то, что записанная ситуация может быть преобразована формально, т. е. с помощью компьютера. Для этого достаточно знать свойства отношений. Например, отношение «быть одновременно» (r_3) обладает свойством симметрии:

$$(A_1 r_3 A_2) \leftrightarrow (A_2 r_3 A_1),$$

т. е. из того, что A_1 одновременно с A_2, следует, что и A_2 одновременно с A_1, А отношение «находиться в» (r_4) обладает свойством транзитивности:

$$\text{если } (A_1 r_4 A_2) \text{ и } (A_2 r_4 A_3), \text{ то } (A_1 r_4 A_3),$$

т. е. если A_1 находится в A_2, а A_2 — в A_3, то A_1 находится в A_3. Как в известной сказке: игла в яйце, яйцо в утке, следовательно, игла в утке — это транзитивное заключение.

Свойством транзитивности обладают многие отношения, но далеко не все. Например, отношение «любить». Если A_1 любит A_2, а A_2 любит A_3, то далеко не всегда A_1 любит A_3. Таких свойств, определяющих возможность преобразования описания ситуации, найдено почти 400. Именно они позволяют трансформировать описание, записанное на этом языке.

Другим способом преобразования полученного описания объекта является образование новых понятий. Под понятием здесь будем подразумевать ситуацию, часто повторяющуюся в описании объекта. Так, «такси, движущееся без пассажира» — ситуация частая и волнующая таксистов. Именно поэтому возникло жаргонное понятие «холостяк», которое определяет описанную ситуацию (не путать с холостым человеком).

Обобщение может происходить по ситуациям. Естественно

обобщать ситуации, которые отличаются лишь понятиями (обобщение по понятиям), отношениями (обобщение по отношениям) и именами (обобщение по именам). Например, при описании работы почты очень важно определить постоянного получателя. Им могут быть отдельные люди, учреждения, заводы, магазины и т. д.— все они потребители корреспонденции. Можно их обобщить одним понятием «абонент». Аналогично происходит обобщение по отношениям и именам.

Таким образом, язык ситуационного управления позволяет не только отобразить смысл фраз не естественном языке в памяти компьютера, но и трансформировать запись, сохраняя ее смысл, и при этом делать обобщения.

При описании технических объектов нет необходимости в особой выразительности языка и все фразы можно свести к простой конструкции (ядерной цепочке):

субъект — акция — объект

где на первой позиции всегда стоит субъект, тот, кто (или что) действует (совершает акцию), на второй — название самого действия (акции), а на третьей позиции объект акции, то, на что воздействует субъект. Например, $(B_1 r B_2)$, где B_1 — манипулятор (это субъект); r — захватил (акция); B_2 — деталь (объект акции).

Такой язык называют универсальным семантическим кодом и применяют для управления роботами, технологическими процессами и т. д. Это, как видно, частный случай языка ситуационного управления.

СЕМАНТИЧЕСКИЕ СЕТИ

Ситуацию, описанную в предыдущем разделе, можно представить в виде графа (рис. 21), вершинами которого являются понятия, имена и их объединения (имен и понятий), а дугами — отношения.

Однако такое представление ситуации не очень удобно: заставляет повторять в графе каждое новое упоминание любого предмета. Так, на рис. 21 дважды фигурирует понятие «конвейер», хотя речь идет об одном и том же предмете. Требование грамматики языка ситуационного управления не позволяет с одним и тем же понятием связать сразу несколько отношений. Это надо делать порознь и связывать их довольно надуманным симметричным отношением «быть одновременно». Более удобное графическое представление — семантические сети. Здесь вершинами графа являются не только понятия (объекты, свойства, состояния) и их имена, но и отношения, а дугами — их связи в виде типа свойств («размер», «цвет», «время», «интенсивность» и т. д.) и так называемых падежей действий — это не что иное, как отношения, уже известные нам, но специального вида.

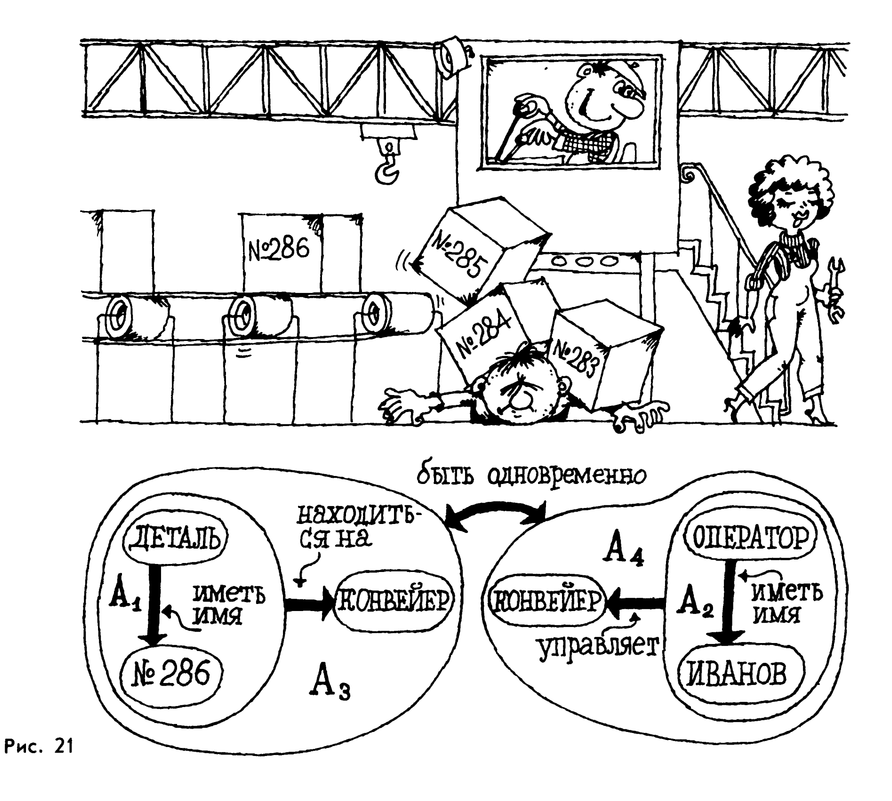


Рис. 21

Не следует путать падежи действия с грамматическими падежами. Падежи действия в отличие от грамматических устанавливают отношение действия между объектами (точнее, их понятиями), указывают, что на что действует. Грамматические падежи устанавливают лишь грамматические отношения между главными и зависимыми словами. Например «прочитать книгу», здесь главным является глагол «прочитать», требующий винительного падежа зависимого слова «книга».

Как видно, вершины семантической сети образуются почти всеми теми словами естественного языка, которые описывают ситуацию, кроме предлогов.

Например, уже знакомая нам ситуация, описанная на языке ситуационного управления и представленная в виде графа на рис. 21, может быть представлена в виде семантической сети (рис. 22). Здесь к уже известным нам отношениям добавляются «иметь имя» и «приемник действия». Отношение «приемник действия» определяет предмет («конвейер»), на который распространяется действие («находиться»). Это отношение близко к отношению «объект действия», но не совпадает с ним.

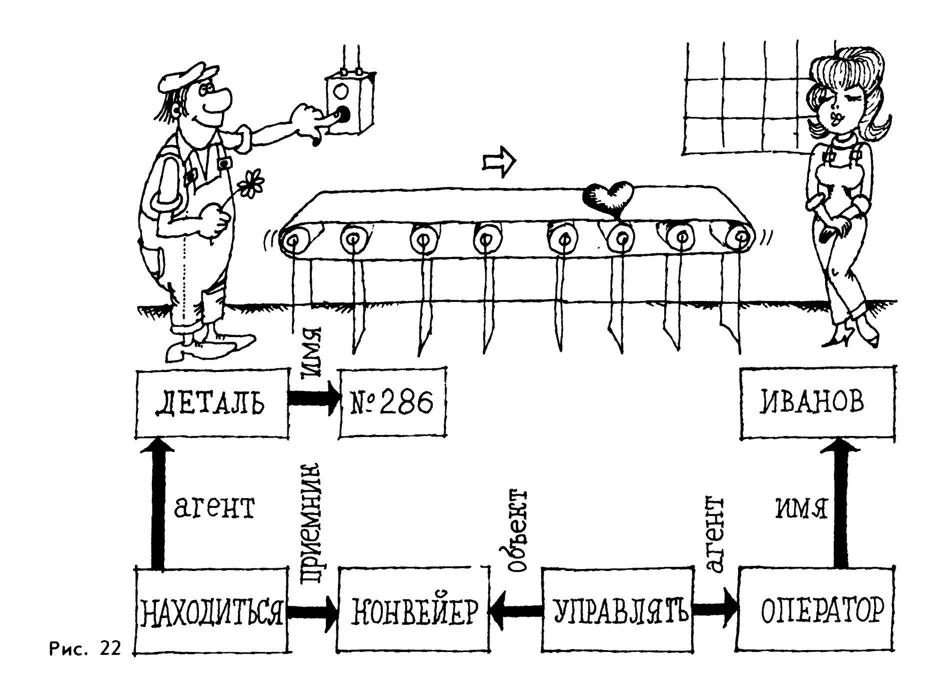


Рис. 22

Как видно, с помощью семантических сетей можно эффективно представлять знания об объектах, ситуациях, явлениях и процессах в памяти компьютера. Для этого достаточно запомнить перечень вершин графа сети и ребра отношений между вершинами. Например, последний пример семантической сети, описывающий ситуацию: «Деталь № 286 находится на конвейере, которым управляет оператор Иванов», запишем в память компьютера в виде следующего графа. Он описывается списком вершин и ребер.

Вершины: 1— ДЕТАЛЬ; 2— № 286; 3— НАХОДИТЬСЯ; 4— КОНВЕЙЕР; 5— УПРАВЛЯТЬ; 6— ОПЕРАТОР; 7— ИВАНОВ.

Это перенумерованный перечень вершин графа семантической сети (дальнейшие ссылки на вершины производятся по их номерам). Ребра обозначаются двумя номерами соединяемых ими вершин:

Ребра: (1, 2), (6, 7) — ИМЯ; (3, 1) (5, 6) — АГЕНТ; (3, 4) — приемник; (5, 4) — объект;

Это перечень всех ребер графа сети с наименованиями отношений, выражаемых этими ребрами. Здесь, например, (1, 2) обозначает ребро, направленное из вершины 1 в вершину 2.

С помощью семантических сетей удобно описывать знания не только о сложившейся ситуации, но и о предметной области, в которой разворачиваются описываемые события, ситуации,

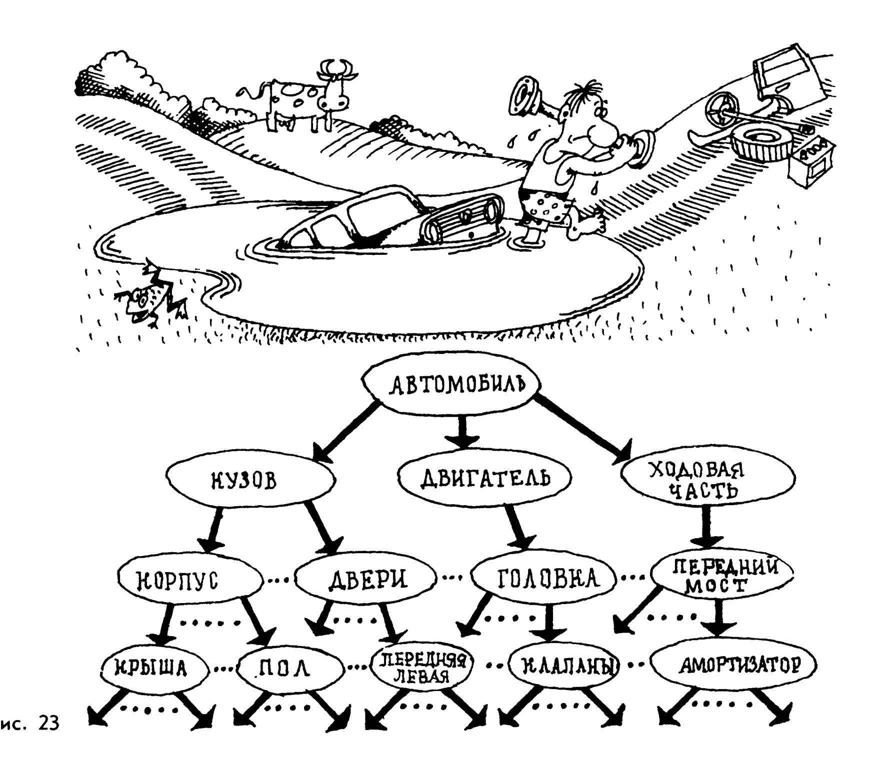


Рис. 23

явления. Знание такой предметной области очень важно компьютеру как для понимания сложившейся ситуации, так и для организации конкретных действий по принципу «Если сложилась ситуация A_i, надо воспользоваться управлением U_i», что на языке ситуационного управления записывается в виде фразы $(A_i r U_i)$, где r — отношение «влечет за собой».

Приведем пример предметной области, описывающей устройство автомобиля. Фрагмент семантической сети этой предметной области приведен на рис. 23. Здесь стрелками обозначено одно отношение «быть частью» — нижний объект является частью верхнего. Располагая в памяти такой сетью, компьютер будет комплектовать конвейеры сборки агрегатов и узлов автомобильного завода. Наличие только одного отношения «быть частью» обедняет эту модель настолько, что часто ее относят не к знанию, а к данным. Такого рода базу данных называют иерархической.

ФРЕЙМЫ

Различные объекты (ситуации, явления и т. п.) описываются различными семантическими сетями — иначе мы бы их не различали. И тем не менее в этих описаниях есть нечто общее,

7 Зак 2074

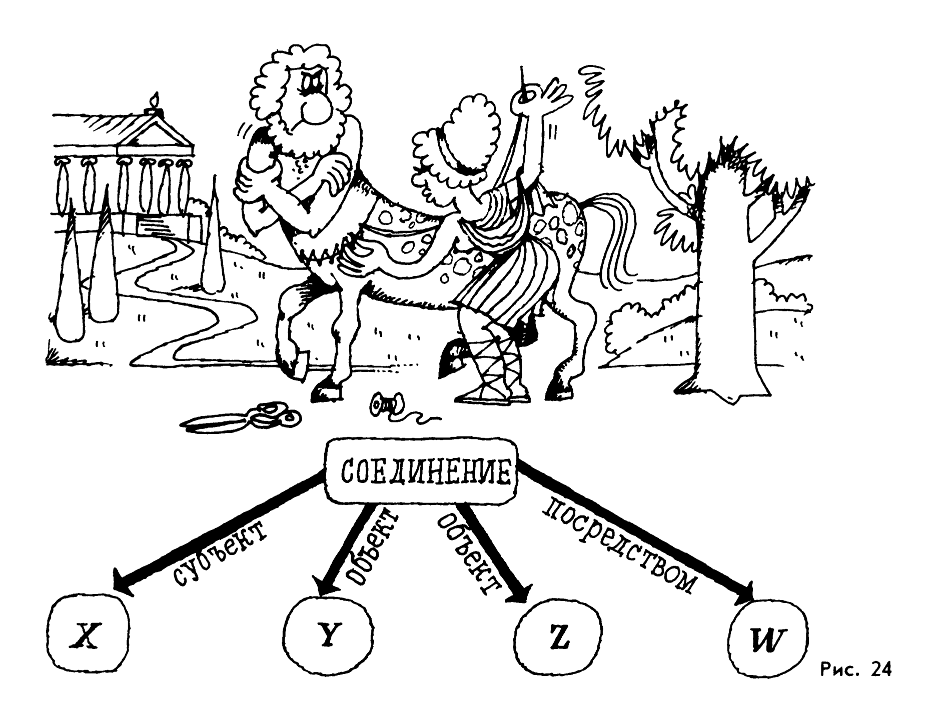


Рис. 24

похожее, стереотипное. Например, действия «любить» и «колотить» неизбежно вызывают целую цепочку естественных вопросов: «что и кого любит (колотит)?», «за что любит (колотит)?», «чем колотит?» и т. д. Очевидно, что любая семантическая сеть, содержащая эти действия, должна иметь ответы на эти вопросы. Это значит, что некоторые части разных сетей имеют одинаковую структуру, описывающую типичную, стереотипную ситуацию (например, ситуации под условными названиями «любовь» и «драка»).

Структура, предназначенная для представления стереотипной ситуации, называется *фреймом* (в переводе с английского языка — рамка, скелет, остов, т. е. то, на чем держится более сложная конструкция). Например, фрейм «встреча» характеризуется следующими необходимыми параметрами: инициатор (т. е. тот, кто назначил эту встречу), участник (тот, кто согласился встретиться), место встречи, время, характер (деловая, дружеская, любовная и т. д.).

Когда все эти параметры определены, то фрейм называют *фреймом-примером*. Это одна из многих встреч, отличительной чертой которой является ее конкретность, т. е. участие в ней конкретных людей, в определенном месте, в заданное время и с известной целью. Это, безусловно, знание и знание факта.

Обобщением многих фактов является *фрейм-прототип*. Неко-

торые параметры этого фрейма не определены полностью и их предстоит еще выявить. Такие неопределенные параметры фрейма называют *слотами* — пустые места, которые необходимо заполнять для функционирования фрейма. Например, фрейм-прототип «соединение» (рис. 24) отражает стереотипную ситуацию: «Субъект X соединяет объект Y с объектом Z способом W». Здесь значения переменных X, Y, Z, W, образующих слоты, сначала не определены и фиксируются, т. е. определяются в процессе взаимодействия фрейма со средой, например в процессе диалога компьютера с пользователем. Таким образом, фрейм-прототип характеризует структуру не полностью определенного знания, которое еще нужно получить и только потом использовать в виде семантической сети. Наличие слотов стимулирует активность компьютера по их заполнению, которое может быть интерпретировано как «познание» компьютером окружающего мира. Действительно, если знания компьютера представляются в виде семантической сети, то заполнение и составление ее естественно назвать компьютерным познанием (без кавычек).

— Ну-ка представьте мне, Поль, версию «Некто отравил Мари цианистым калием» в виде семантической сети.

— Очень просто. Здесь четыре вершины сети будут соответствовать четырем понятиям: «некто», «Мари», «цианистный калий» и «отравлять». Обозначим их цифрами 1, 2, 3 и 4 соответственно. Связи между этими вершинами определяются падежами действия (отношениями): «быть объектом действия», «быть агентом действия» и «быть средством действия», т. е. посредством чего совершается действие. В результате получаем такую семантическую сеть (рис. 25, *а*).

— А как эта версия будет записана в памяти машины?

— Очено просто:

вершины: 1— НЕКТО; 2— МАРИ; 3— ЦИАНИСТЫЙ КАЛИЙ; 4— ОТРАВЛЯТЬ

ребра: (4, 2),— ОБЪЕКТ; (4, 1) — АГЕНТ; (4, 3) — СРЕДСТВО.

Здесь словами ОБЪЕКТ, АГЕНТ и СРЕДСТВО обозначены указанные выше падежи действия.

— Да, но в каждом отравлении есть объект, агент и средство, так стоит ли об этом говорить?

— Правильно. Надо в память машины ввести фрейм «отравление» (рис. 25, *б*), где слоты X, Y, Z могут заполняться определенными значениями в каждом конкретном случае. У нас: X=некто, Y=Мари, Z=цианистый калий.

— Но «некто» — это не определенное лицо. Это скорее слот, так как X не определен? — спросил Мегрэ.

— Вы правы, шеф. Это лишь фрейм-прототип, где значение X еще не определено. А фреймом-примером он станет лишь тогда, когда мы узнаем, кто это был.

А если Мари отравилась сама?

— Тогда X=Y=Мари и наш фрейм-пример имеет смысл «Мари отравилась»,

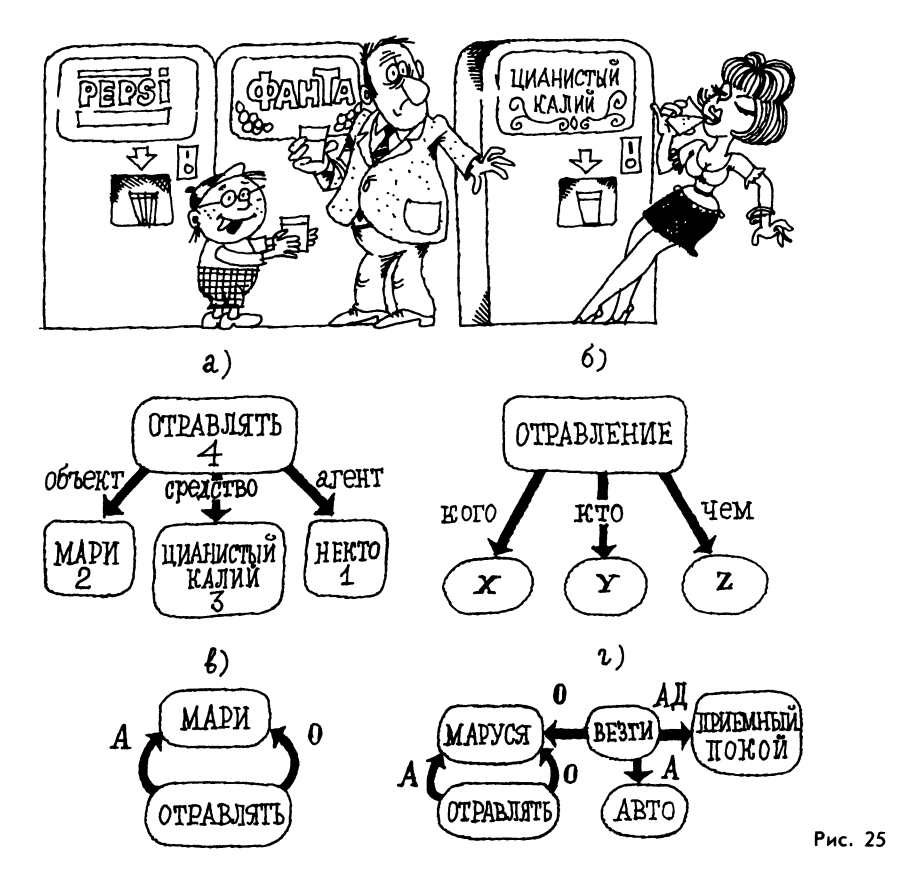


Рис. 25

то же самое можно записать в виде семантической сети (рис. 25, *в*), где Мари является и агентом (*А*), и объектом (*О*) действия.

— Если вы, Поль, так ловко записываете ситуацию, то представьте в виде сети слова из известной русской песенки «Маруся отравилась, везут в прием-покой».

— Для представления этой грустной ситуации не хватает данных, очень важных для компьютера: кто везет Марусю?

— Ну, положим, автомашина.

— Тогда семантическая сеть будет такой (рис. 25, *г*). Здесь к известным падежам (отношениям) действия *А* и *О* (быть агентом и объектом) добавился третий: *АД* — «быть агентом действия», который связывает понятия «везти» и «прием-покой», т. е. определяет, куда направлено действие. Эта сеть интерпретируется так: «Маруся отравилась, авто везет ее в прием-покой». Смысл, как видите, сохранен, а «поэзия» (увы!) потерялась. Так бывает очень часто, когда мы «поверяем алгеброй гармонию».

— Бог с ней, с гармонией,— согласился Мегрэ.— Меня она в данный момент не волнует. Но что из того, что компьютер теперь что-то знает. Ведь

и мы это знаем. И ничего нового при этом не произошло! Никакой новой для нас, людей, информации. Зачем все это нужно нам, сыщикам?

— Конечно, из того факта, что информация введена в компьютер, ничего не следует. При этом мы не только не приобретаем, но и теряем информацию (известно, что при всяком формальном преобразовании, а представление ситуации в виде семантической сети является формальной процедурой, информация может только потеряться, как потерялась поэзия в нашем примере с Марусей). Эта потеря неизбежна. Но при этом открываются две возможности, которых не было ранее.

Во-первых, если описываемая ситуация очень сложная, ее семантическая сеть имеет сотни или тысячи вершин-понятий, разобраться «старым способом» в ней очень трудно. Именно здесь может помочь компьютер. Например, имеем сеть различных знакомств большого числа людей с различными отношениями: «знаком», «хорошо знаком», «близко знаком», «дружит» и т. д. Как узнать возможности общения в такой группе? Через каких лиц может один из них обратиться к другому, незнакомому ему, с деликатной просьбой, передача которой возможна только по ребрам «близко знаком» или «дружит» в графе этой группы? Если группа мала, то это легко определить и без компьютера. А если это целый город? В этом случае без компьютера не обойтись. Но предварительно в его память следует ввести граф знакомств этой группы.

Во-вторых, формальность семантической сети позволяет манипулировать с ней формальными средствами. Например, как изменится ситуация, если предпринять какое-то действие. Ведь семантическая сеть не более чем модель, описывающая сложившуюся ситуацию. А располагая моделью, мы всегда можем определить (точнее, вычислить) ее реакцию на то или иное воздействие. И, если модель адекватна объекту, так же поведет себя и реальная ситуация при этих воздействиях. Более того, желая получить какую-то конкретную ситуацию, вы можете потребовать у компьютера рекомендаций, как поступить, чтобы получить то, что вы желаете, т. е. перевести исходную ситуацию в целевую.

— Исходная ситуация вам известна не хуже меня, Поль, а целевая — знать преступника. Что предложит вам компьютер?

— Пока ничего,— засмеялся Поль,— его еще следует начинить необходимой информацией, ввести в его память семантическую сеть сложившейся ситуации.

— Вот и займитесь этим, дружище.

— Слушаю, комиссар,— ответил Поль и проворчал что-то об энтузиастах, чья инициатива всегда наказывается.

13. ИНТЕЛЛЕКТУАЛЬНЫЙ ЯЗЫК

(Языки искусственного интеллекта)

ПРОЦЕДУРНЫЕ И ДЕКЛАРАТИВНЫЕ ЗНАНИЯ

Процедурное знание — это знание о том, как достичь каких-то целей, знание процедур, позволяющих добиваться заданных целей, а декларативное — это знание о том, что есть что, т. е. декларация, указывающая «что это такое?». Например,

умение изготовить салат или сложить два числа является процедурным знанием, а сведения о том, что такое «салат» или что такое «сложение», являются декларативными. В алгоритмических языках всегда необходимо прибегать к декларации при определении переменных, например X = integer означает, что X — целое число, а Y = real, что Y — действительное.

Для описания знаний, необходимых для «интеллектуальной» работы компьютера, разработаны специальные языки, позволяющие представлять различные аспекты знания — факты, отношения, процедуры и т. д. Эти языки отличаются от алгоритмических (они рассмотрены в гл. 5) тем, что имеют средства для декларативного описания объектов. Если процедурные языки (Фортран, Алгол, Паскаль и т. д.) позволяют описывать преимущественно процедуры, то декларативные языки — объекты. Рассмотрим наиболее типичный процедурный язык.

ПРОЛОГ

Этот язык был реализован в 1971 г. в Марсельском университете (Франция) для логического программирования (PROLOG = PROgramming LOCig). Теоретическую основу этого языка разработал Р. Ковальский. Программирование на языке Пролог состоит из трех действий:

задание фактов, относящихся к объектам, и отношений между объектами (это по сути дела база данных, с помощью которой описываются объекты и их отношения);

задание правил манипулирования объектами и отношениями;

постановка вопроса относительно объектов и их отношений.

Начнем с одночленного отношения «объект a имеет свойство b». Оно записывается на Прологе в виде $b(a)$. Например, факт «Пушкин — русский» записывается в виде русский (пушкин), т. е. объект «Пушкин» (это объект a) имеет свойство «быть русским» (это свойство b). Отметим, что прописные буквы в Прологе используются только для переменных, а конкретные имена, даже собственные, пишутся строчными буквами, что непривычно.

Двучленное отношение «объект a имеет свойство b по отношению к объекту c», определяющее отношение между двумя объектами, записывается в виде $b(a, c)$. Факт «Пушкин — автор „Евгения Онегина“», являющийся отношением авторства между Пушкиным и его произведением, записывается так: автор (пушкин, евгений онегин), т. е. объект «Пушкин» по отношению к объекту «Евгений Онегин» имеет свойство «быть автором».

Теперь можно записать следующую программу на Прологе:

```
русский (пушкин)
англичанин (шекспир)
```

автор (пушкин, евгений онегин)
автор (шекспир, отелло)

В этой программе декларируются следующие известные факты: Пушкин — русский, а Шекспир — англичанин, Пушкин — автор «Евгения Онегина», а Шекспир написал «Отелло». Эта программа является базой данных, а не знаний, как это может показаться с первого взгляда. Действительно, для нас, людей, это база знаний, так как нам известно, кто такой был Пушкин, что такое иметь национальность и что значит быть автором какого-то произведения. Для компьютера же это не более чем какие-то слова, расположенные в определенном порядке, т. е. данные, смысл которых неизвестен компьютеру.

Итак, имеем базу данных. Чтобы воспользоваться этими данными, необходимо иметь возможность получать ответы на вопросы, возникающие по поводу информации, записанной в базе данных. В Прологе такая возможность есть.

Вопрос в Прологе начинается со знака вопроса. Например, вопрос «Какой объект имеет по отношению к объекту c свойство b?» (здесь c и b — конкретные имена объекта и свойства) записывается в виде

? — $b(X, c)$

где X обозначает искомое неизвестное переменное. Например, вопрос «Кто автор „Отелло"?» (или, точнее, кто имеет свойство «быть автором» по отношению к объекту «Отелло»?) программируется на Прологе так:

? — автор (X, отелло)

Ответ компьютера на этот вопрос:

X = шекспир

А в ответ на вопрос «является ли Шекспир автором «Евгения Онегина»? или на Прологе:

? — автор (шекспир, евгений онегин)

получим ответ «нет».

Если хотим узнать, что написал Пушкин, следует задать вопрос в виде

? — автор, (пушкин, X)

На что получим ответ:

X = евгений онегин

Не следует удивляться такой односложности ответа — в базу данных, к которой мы обращались, было заложено лишь название этого романа. Если бы базу данных расширить сведениями о дру-

гих произведениях Пушкина, то компьютер выдал бы все, чем располагает.

Вопросы в Прологе можно задавать только относительно объектов, а не отношений. Так, интересуясь национальностью автора «Отелло», нельзя задать вопрос

? — $X(Y)$, автор (Y, отелло)

здесь переменной X является отношение, что запрещено в Прологе.

Чтобы все-таки получать ответы на такие вопросы, нужно искомое отношение сделать объектом — вместо отношения «иметь национальность» ввести объект «национальность». Для этого удобно ввести отношение «явл» (является):

явл(a, b)

которое выражает факт, что «a является b». Например, факт «Тургенев — русский» (точнее, «Тургенев является русским») можно записать в виде

явл (тургенев, русский)
Теперь можно записать следующие факты на Прологе:
явл (пушкин, русский)
явл (шекспир, англичанин)
явл (тургенев, русский)
явл (диккенс, англичанин)
автор (пушкин, евгений онегин)
автор (тургенев, отцы и дети)
автор (шекспир, отелло)

Можно задать вопрос о национальности автора «Евгения Онегина»:

? — автор (X, евгений онегин), явл (X, Y)

Здесь пришлось ввести дополнительную неизвестную X (фамилия автора), которая нас не интересует, но нужна для формулировки вопроса. В результате получим избыточный ответ:

X = пушкин
Y = русский

Если хотим задать вопросы о соотечественниках, следует ввести отношение «соот» (соотечественник):

соот (a, b)

которое означает, что a является соотечественником b, и дополнить базу данных очевидными фактами:

соот (пушкин, тургенев)
соот (шекспир, диккенс).

Но можно поступить по-другому. Определить отношение соот

(X, Y) через уже заданные. Действительно, если два человека имеют одну национальность, то они соотечественники и поэтому из того, что

явл (X, Z), явл (Y, Z)

т. е. X и Y имеют одну национальность Z, следует, что X и Y — соотечественники;

соот (X, Y)

Следовательно, это отношение связано с отношением «явл». Эта связь в Прологе записывается следующим образом:

соот (X, Y): — явл (X, Z), явл (Y, Z)

(Пусть читатель не придирается к этому не вполне строгому определению, которое, вообще говоря, расходится с понятием соотечественника как лица, имеющего с кем-то одно отечество, а не национальность.)

Теперь к базе данных, дополненной этим отношением, можно обратиться с вопросами, требующими логического вывода:

? — соот (пушкин, шекспир)
? — соот (пушкин, тургенев)

Являются ли Пушкин и Шекспир соотечественниками? А Пушкин и Тургенев? Ответы на эти вопросы выводятся путем соответствующих логических преобразований. На первый будет получен ответ «нет», а на второй «да».

Как видно, на Прологе легко создать базу данных в виде отношений между объектами, вводить новые отношения и задавать любые вопросы, ответы на которые могут содержаться в базе данных. При этом не нужно программировать сам процесс вывода ответа — он уже заложен в языке Пролог.

А вот вычислять на Прологе очень сложно: он для этого не предназначен. Например, чтобы сложить два целых числа, x и y, и получить их сумму $z=x+y$, надо ввести трехчленное отношение «сум» — суммирование в виде

сум (X, Y, Z)

и записать в базу данных все возможные варианты суммы, которые могут встретиться в работе:

сум (1, 1, 2)
сум (1, 2, 3)
сум (2, 2, 4)
....

Теперь к компьютеру можно обращаться с вопросами о сумме двух чисел:

? — сум (a, b, X)

и получать ответ: $X=3$, если $a=1$ и $b=2$. Но, например, на вопрос:

? — сум $(2, 1, x)$

будет ответ «нет», если отношение сум (2, 1, 3) не заложено в базу. Если же правило коммутативности сложения $x+y=y+x$ заложить в базу в виде определения

сум (X,Y,Z) : — сум $(Y,X,Z,)$

то компьютер справится с этой задачей.

Конечно же, это очень неудобно, и при большом количестве вычислений Пролог расширяют специальными вычислительными процедурами типа тех, которые используются в процедурных языках, например в Фортране.

Рассмотрим примеры правил преобразования данных. Пусть объектами являются аналитические выражения, например полиномы. Тогда можно ввести отношение упр (X,Y), которое означает, что выражение Y является упрощением выражения X, что и позволяет упрощать аналитические выражения. Но для этого нужно определить дополнительные правила манипулирования. Например, так:

упр $(X*1, X)$
упр $(1*X, X)$
упр $(X*0, 0)$
упр $(0*X, 0)$

что выражает очевидные соотношения: $X*1 = 1*X = X$ и $X*0 = 0*X = 0$. Теперь, если в процессе вывода встретится выражение $X*1$, оно будет немедленно заменено на X, т. е. упрощено.

ЛИСП

Этот язык в начале 60-х годов был разработан в США под руководством известного ученого Дж. Маккарти специально для обработки символьной информации, в частности текстов на естественном языке. Например, для управления сложной системой, где информация задается фразами на английском языке. Лисп нашел свое применение для решения логических задач и в настоящее время широко используется для решения задач искусственного интеллекта.

Лисп является языком обработки списков. Под *списком* понимается любой текст (в широком смысле), состоящий из атомов и подсписков. Например, *атомами* являются выражения ALMA, 523, СТОЛ, D35TZ и т. д., которые не могут быть разделены на

свои части — на то они и атомы. *Подсписком* является любая часть списка, состоящая из атомов или более мелких подсписков. Все списки и подсписки в отличие от атомов заключаются в круглые скобки. Например:

(КОМПЬЮТЕР РЕШАЕТ ЗАДАЧУ Д36)
(THIS IS(A LIST)
(ПОЛОЖИ (КАРАНДАШ) (НА СТОЛ))

Первый список состоит из одного подсписка — самого себя или из четырех атомов, второй — из двух атомов и подсписка, а третий — из двух подсписков (карандаш) и (на (стол)) и одного атома (положи); второй подсписок состоит из двух атомов (на) и (стол).

В языке Лисп роль операторов выполняют функции, с помощью которых и происходит обработка символьной информации. Приведем примеры наиболее распространенных функций Лиспа.

Функция CAR(X) выделяет первый элемент списка (X), к которому она применяется, например если (X) = (ДЕТАЛИ ПОДАЮТСЯ НА КОНВЕЙЕР), то

(CAR(X)) = (ДЕТАЛИ)

— это и есть значение, которое принимает функция CAR на списке X.

Функция CDR(X) выделяет остаток списка (X), которое получается при вычеркивании первого элемента списка (X). Продолжая пример того же списка (X), получаем:

CDR(X) = (ПОДАЮТСЯ НА КОНВЕЙЕР).

Как видно, обе эти функции дополняют друг друга.

Функция CONS объединяет списки (X) и (Y) в один. Например, если (X) = (ДЕТАЛИ), (Y) = (В СБОРКЕ), то

(CONS X Y) = (ДЕТАЛИ В СБОРКЕ)

Частным случаем символа является число. Для обработки числовой информации в Лиспе есть специальные функции. Так, операции сложения и умножения реализуются функциями PLUS и TIMES. Например, выражение (PLUS X Y) дает сумму $X + Y$, а (TIMES X Y) — произведение $X \cdot Y$.

Переменные и константы в Лиспе являются атомами. Роль оператора присваивания здесь играет функция SETQ. Например, выражение

(SETQ X 8)

присваивает переменной X значение 8, т. е. получаем $X = 8$.

Программа на Лиспе выполняется последовательно, в порядке следования записей, который может нарушаться оператором безусловного перехода по метке (DOM), где M — метка функ-

ции, к которой следует перейти (как в Фортране к оператору), и функцией условного перехода (COND X Y). Эта функция имеет в качестве аргументов два списка. В первом (X) формулируются условия, которые необходимо проверить. А во втором (Y) описывается то, что следует сделать, если эти условия выполнены (при их невыполнении следует обращаться к следующей в программе функции).

Проиллюстрируем, как на языке Лисп может быть определена сумма заданных чисел. Эти числа в памяти хранятся в виде списка, например

$$(X) = (56\ \ 83\ \ 0{,}37\ \ 17)$$

Напишем Лисп-программу суммирования и прокомментируем ее:

(SETQ S0)	$S=0$
A (SETQ S (PLUS S(CAR X)))	$S:=S+$(CAR X) — прибавление к S первого элемента (атома) списка (X). A — метка
(SETQ X(CDR X))	Список (X) превращается в список, укороченный на первый атом
(COND ((NULL X)(GO B)))	Если список (X) пустой, т. е. $X=$ $=$NULL, то перейти по метке B, в противном случае — к следующему оператору
(GO A)	Перейти по метке A
B (PRINT S)	Печатать значение S. B — метка.

Здесь NULL означает пустое множество; если (NULL X) $= T$ (истина), то (X) — пустой список.

Приведем примеры других арифметических функций Лиспа (их обычно называют примитивами):

ABSVAL — вычисление абсолютного значения;

ENTIER — вычисление целой части выражения;

EXPT — вычисление значения одного аргумента, возведенного в степень другого аргумента;

RECIP — вычисление числа, обратного аргументу;

REMAINDER — вычисление остатка от деления двух аргументов (первый на второй);

MAX — вычисление наибольшего из группы (списка) аргументов.

В Лиспе имеются операции над множествами. Так, оператор INTERSECTION выдает список всех элементов, общих для двух множеств, т. е. реализует операцию пересечения этих множеств (списков), а UNION — список элементов, каждый из которых принадлежит хотя бы одному из двух множеств (списков), т. е.

реализует операцию объединения. Первый элемент списка в Лиспе интерпретируется как имя функции, а остальная его часть — как множество аргументов этой функции. Например, список (EXPT *X* *Y*) вырабатывает число, равное X^Y, а (MAX 5 8 13 7) — число 13. Поэтому Лисп часто называют языком функций. Это означает, что каждая конструкция в Лиспе записывается и выполняется как функция, аргументами которой могут быть любые списки.

Лисп-программа представляет собой описание функций, определяемых пользователем через стандартные функции Лиспа (примитивы). Выполнение программы заключается в определении значений введенных функций, результатом чего может быть тоже список.

В заключение отметим, что и Пролог, и Лисп являются признанными языками искуственного интеллекта, позволяющими решать невычислительные задачи, свойственные искусственному интеллекту. Трудно сказать, какой из них лучше для этих целей. Скорее всего для одних задач лучше — один, а для других — другой. Но так как Лисп родился в США, а Пролог — в Европе, то это и определило их преимущественное применение: Пролог в Европе (и Японии), а Лисп в США для любых задач искусственного интеллекта.

— Откровенно говоря, — задумчиво сказал Мегрэ, — мне кажется, что эти языки искусственного интеллекта не очень-то интеллектуальные. Они лишь позволяют обрабатывать информацию, представленную в виде фраз естественного языка, т. е. преобразовывать их из одной формы в другую. От интеллектуального языка я ждал большего. Ну, например, возможности выявлять новые связи, факты, обстоятельства — все то, что было неизвестно. Не так ли, Поль?

— Вы требуете невозможного, шеф, — улыбнулся Поль. — Ведь любой алгоритмический язык — это не более чем формальная система, способная лишь преобразовать информацию из одной формы в другую. И если в исходном тексте не было нужной вам новой информации, то никаким преобразованием этого текста ее получить нельзя. Совсем другое дело, если ситуация, описываемая этим текстом, очень сложна и при беглом знакомстве не позволяет получить то, что нужно и содержится в тексте. Именно для решения такого рода задач и придуманы языки искусственного интеллекта.

— Ну, что ж! — согласился Мегрэ, — именно это мне и нужно. Вот вам исходные тексты, — и он указал на кипы сообщений, полученных со всего света. — Покажите мне, как можно использовать язык искусственного интеллекта, чтобы получить (точнее, выявить) нужную мне информацию. Она там содержится, уверяю вас.

— Все не так просто, — грустно заметил Поль. — Ваши тексты записаны на естественном языке со всеми вытекающими отсюда неприятными последствиями.*

* Они описаны выше, в гл. 10.

— Но ведь языки искусственного интеллекта обрабатывают именно тексты, записанные на естественном языке, — удивился Мегрэ, — какие же «неприятные последствия» могут возникнуть?

— Здесь перед нами довольно неприятная дилемма. Если мы хотим формально работать с текстом на естественном языке, обрабатывать его с помощью языков искусственного интеллекта, то, чтобы преодолеть все трудности, связанные с его естественностью, нужно исходный текст преобразовать к формальному виду, с которым и будет работать программа. Так это делается при работе с Прологом. Например, чтобы факт «Мари отравилась» занести в базу данных (а это и есть формальная запись) необходимо ввести двухместное отношение «быть отравленным»:

быть отравленным (a, b)

что означает «a отравил b». Тогда факт самоотравления Мари можно записать в виде «быть отравленным (мари, мари)». К сожалению, с помощью компьютера этого преобразования сделать нельзя. Здесь нужен человек, понимающий текст и владеющий приемами представления фактов в виде так называемых *предикатов* — это и есть те самые отношения, с которыми работает Пролог.

— А нельзя ли все-таки обойтись без предварительного неформального преобразования текста? — спросил Мегрэ и уныло взглянул на громадные кипы сообщений.

— Можно, это допускает Лисп. Но результат будет не очень интересен для того, кто хочет извлечь новую информацию. Например, с помощью Лиспа можно легко построить словарь текста, т. е. расположить все его слова в алфавитном порядке с указанием, сколько раз каждое слово встречалось в данном тексте. Эту задачу приходится часто решать при идентификации текста, т. е. отыскании его автора.

— Мне в данном случае интереснее Пролог. Выходит, что для работы с этим языком я должен сам ввести все отношения, которые есть в моем тексте, и переписать его в виде набора таких отношений?

— Да, хотя нельзя строго сказать, что в тексте априори есть определенные отношения. Это вы сами их вводите в зависимости от того, как понимаете текст.

— Выходит, что один и тот же текст я могу записать в предикатах по-разному? — изумился Мегрэ.

— Конечно! Например, если у вас нет сомнений, что Мари отравилась (сама), то естественно ввести отношение «отравиться». Это одноместный предикат вида

отравиться (a)

где a — имя того, кто отравился. Если же сомнения есть, то нужно использовать двухместный предикат «быть отравленным», который допускает возможность не только самоотравления.

— Ну, хорошо, — согласился Мегрэ. — А какие еще предикаты следует ввести для расследования нашего убийства или самоубийства?

— Их много. Например, можем всех персонажей нашего дела описать четырехместным отношением «персона» вида

персона (имя, возраст, пол, специальность)

Например, несчастную Мари в виде персона (мари, 18, женск., продавщица). Важен мотив убийства, если оно было. Тогда нужно ввести одноместный предикат «мотив» в виде

мотив (*a*),

где *a* может принимать значения: *a* = ревность, деньги, страх и т. д. Для описания любовной интриги естественно ввести отношение «иметь любовную связь». Очевидно, что оно двухместное:

иметь любовную связь (*a*, *b*)

где *a* и *b* — имена любовников. Чтобы описать орудие убийства и его владельца, если он не определен точно, нужно ввести двухместное отношение «вероятно владеет» в виде

вероятно владеет (*a*, *b*)

где *a* — имя того, кто владеет, а *b* — название орудия убийства (в нашем случае это цианистый калий)

Если есть сомнения относительно различных вариантов орудия убийства, то нужно вводить предикат идентичности их действия:

идентично действует (*a*, *b*)

где *a* и *b* — названия орудий, действие которых одинаково, например пистолет и автомат, которые имеют одинаковые пули.

— Хватит, достаточно, — улыбнулся Мегрэ, — мне все ясно. Нужно подробнейшим образом описать все взаимодействия всех возможных участников изучаемого преступления и все предметы, зафиксированные при этом и имеющие отношение к преступлению. Далее ввести предикат «быть убийцей» в виде

убийца (*X*)

где *X* — неизвестное имя убийцы из числа фигурирующих в деле «персон» и задать вопрос:

? — убийца (*X*)

— Не торопитесь, шеф. Если это отношение (быть убийцей) не связано с описанием ситуации, то ответа вы не получите. Нужно связать его с уликами, оставленными на месте преступления. Например, у убийцы 43-й размер обуви и он курит сигареты Кемел, что должно быть отражено введением отношений «курит» и «имеет размер обуви», а также расширить отношение «персона» данными о марке любимых сигарет и размере обуви:

персона (имя, возраст, пол, специальность, курит сигареты, размер обуви)

Именно эти отношения свяжут имя неизвестного убийцы с известными обстоятельствами, например, так:

персона (*X*, *Y*, мужск., *Z*, кемел, 43)

Только после этого можно будет задать ваш вопрос.

— И что, всегда будет получен ответ? — скептически спросил Мегрэ.

— Да, но он может быть различным. «Нет» — будет означать, что имя убийцы не выводится из исходных данных. Несколько имен в ответе означает, что исходных данных мало ...

— А если имя одно, то это и есть наверняка имя убийцы?

— Не всегда. Ведь все зависит от истинности исходной информации. Здесь Пролог может ошибиться так же, как ошибается опытный следователь, если исходит из неправильных предпосылок. Это ведь, по сути дела, не ошибка, а неверная интерпретация исходных данных. Разные решения могут быть приняты Прологом и следователем при одинаковых исходных данных лишь в том случае, если следователь делает ошибочные заключения. Пролог же никогда не ошибается, если, разумеется, не ошибается компьютер, на котором реализуется программа Пролога.

— Неужели Вы, Поль, серьезно считаете, что Пролог действительно может заменить сыщика? — изумился Мегрэ.

— Ну, наверное, пока нет. И все потому, что сыщик располагает некоторой дополнительной информацией, которая пока недоступна машине. Это интуиция, предыдущий опыт, подозрения всякого рода и другая человеческая «дребедень», которую пока неизвестно, как вводить в память компьютера.

— Ага, — засмеялся Мегрэ, — вот вы и дали трещину, несчастный масленщик. Вы уже признаете, что сейчас пока не все человеческие функции можно передать компьютеру. А я уверен, что и всегда в мышлении человека найдется место, некомпьютеризуемое (простите за это ужасное слово). Здесь я опираюсь не на свою уверенность, в которую вы не поверите, а на известную теорему Гёделя. Вы ведь наверняка знаете: в любой формальной системе можно сформулировать утверждение, которое нельзя ни подтвердить, ни опровергнуть, не выходя за рамки этой формальной системы. Так что компьютерное «мышление» всегда ограничено этой теоремой. Именно этим, например, отличается человек, будучи неформальной системой. И его решения, хотя и могут быть ошибочными в силу несовершества процесса мышления, но оно не ограничено по своим возможностям.

— Ну, что вы, шеф, — обиделся Поль, — речь ведь идет не о применении Пролога в следственной практике. Этого никто и не пытается делать. А сказанное лишь иллюстрация к языку, который нашел широкое применение для решения задач искусственного интеллекта.

— Для чего же в действительности применяется Пролог? — спросил Мегрэ.

— Ну, например, для создания интеллектуального интерфейса для прикладных программ.

— Что это такое?

— Прикладная программа, как известно, решает конкретную (прикладную) задачу пользователя. Например, имеются две пули, выпущенные из одного и того же пистолета. Нужно выяснить, которая из них была первой. Это, как известно, очень важно для выяснения, соблюдал ли полицейский инструкцию: стрелять сначала в воздух и только потом, если нападающий не остановился, в него. Вот именно эта, вынутая из тела пуля, должна быть второй, а не первой, как это бывает часто с перепуганными полицейскими.

— А как можно различить эти пули? — спросил Мегрэ.

— Вы знаете, что ствол пистолета смазан, и на первой пуле всегда нагар от

масла чуть больше, чем на второй. Но эта разница столь назначительна, что уловить ее может лишь тщательный анализ на компьютере. Программа этого анализа и есть та самая прикладная программа, с которой вам придется иметь дело, решая эту задачу. Но чтобы она заработала, нужно ввести исходные данные о пулях. Для этого следует выполнить ряд довольно сложных и нудных операций с компьютером. Именно эту неприятную функцию выполняет программа интеллектуального интерфейса. Она находится как бы между вами и прикладной программой, именно поэтому ее и называют интерфейсом. Интеллектуальность ее в том, что она понимает вас — непрограммирующего пользователя, который общается с ней на языке, очень близком к естественному. Но для того, чтобы понимать, она должна иметь модель той предметной области, в которой решается задача. Ведь именно на языке этой предметной области и происходит общение пользователя с компьютером. В примере с пулями эта предметная область определяется понятиями: пуля, первая, нагар и многими другими, а также отношениями между ними. Именно это и записывается в виде предикатов на языке Пролог. Так что программа интерфейса становится интеллектуальной, только если есть модель предметной области, для описания которой и используется язык Пролог.

— Что ж, это понятно. И в подтверждение того, что я действительно понял, позвольте высказаться на Прологе:

быть благодарным (мегрэ, поль)

где я ввел очевидный двухместный предикат благодарности.

14. К КОМПЬЮТЕРУ ЗА КОНСУЛЬТАЦИЕЙ

(Экспертные системы)

ЗАЧЕМ НУЖНЫ ЭКСПЕРТЫ

Древние не без основания считали, что есть только три источника знания: опыт, интуиция и мнение умных людей (сейчас их называют экспертами). В наше время нового источника знаний так и не появилось. Все тот же опыт — его стали чаще называть экспериментом. Все та же интуиция (ее раньше называли божьим промыслом, а на деле это тот же опыт, но отраженный в нашем подсознании). И все то же мнение экспертов, но оно стало более многогранным: появились средства хранения экспертной мудрости — книги. До книг, т. е. до изобретения письменности, знания экспертов передавались устно — в виде притч, легенд, мифов ... Теперь, чтобы узнать мнение эксперта по тому или иному вопросу, редко обращаются к реальным людям — экспертам: найти его обычно труднее, чем нужную книгу.

Но знания эксперта далеко не всегда можно отразить на бумаге. Особенно это касается так называемых плохо структурированных знаний. К такого рода знаниям относятся, например, специальные знания в узкой области, где очень важна интуиция специалиста. Интуитивные знания нужны в любой профессиональной области. И носители таких знаний являются обычно только

эксперты — специалисты в данной конкретной профессиональной области. Именно такого рода знания необходимы для эффективной работы.

Все специалисты хорошо знают это и никогда не упустят возможности поговорить с коллегой, которого считают экспертом в своей области. Такой разговор дает больше, чем прочтение многих монографий. Именно поэтому специалисты так стремятся к личному общению на всякого рода конференциях, совещаниях, симпозиумах, семинарах. Только при таком общении может быть получена та самая «экспертная» информация, которая нужна специалисту для решения конкретных профессиональных задач и которой нет (возможно, пока) в учебниках, справочниках, статьях и т. д. И именно такой информацией следует «начинить» память компьютера, чтобы получить экспертную систему — так называют систему, к которой можно обращаться для получения экспертной информации.

Конкретные экспертные системы появились в результате развития систем искусственного интеллекта и под давлением острой необходимости, связанной с нехваткой специалистов-экспертов, которые смогли бы в любой момент квалифицированно ответить на многочисленные вопросы из своей области. Хороший эксперт всегда малодоступен, а очень хороший тем более. Поэтому так важно и нужно иметь интеллектуальную компьютерную систему, обладающую знаниями эксперта, к которой можно обратиться в любой момент с профессиональным вопросом на естественном языке.

Чтобы понять, почему современному компьютеру просто необходимо стать интеллектуальным, рассмотрим взаимодействие человека с ЭВМ в процессе решения какой-то его конкретной задачи. Напомним, что в таком качестве этого человека называют конечным пользователем.

ПОРТРЕТ КОНЕЧНОГО ПОЛЬЗОВАТЕЛЯ

Легко представить, что собой представляет конечный пользователь. Вот портрет его знаний (точнее, незнаний):

Он НЕ ЗНАЕТ, как устроен компьютер.

Он НЕ ЗНАЕТ, как писать программы для работы с компьютером.

Он НЕ ЗНАЕТ формальных (математических) методов решения задач в его области, что необходимо для использования компьютера. Более того, всего этого он и НЕ ХОЧЕТ УЗНАВАТЬ!

Чтобы читатель не подумал, что конечный пользователь лентяй и лоботряс, напомним, что он специалист в своей области и решает свою конкретную задачу: проектирует, лечит, ищет неисправность в схеме, синтезирует нужные вещества, создает новую технологию, ищет месторождения и т. д. А ЭВМ ему нужна лишь

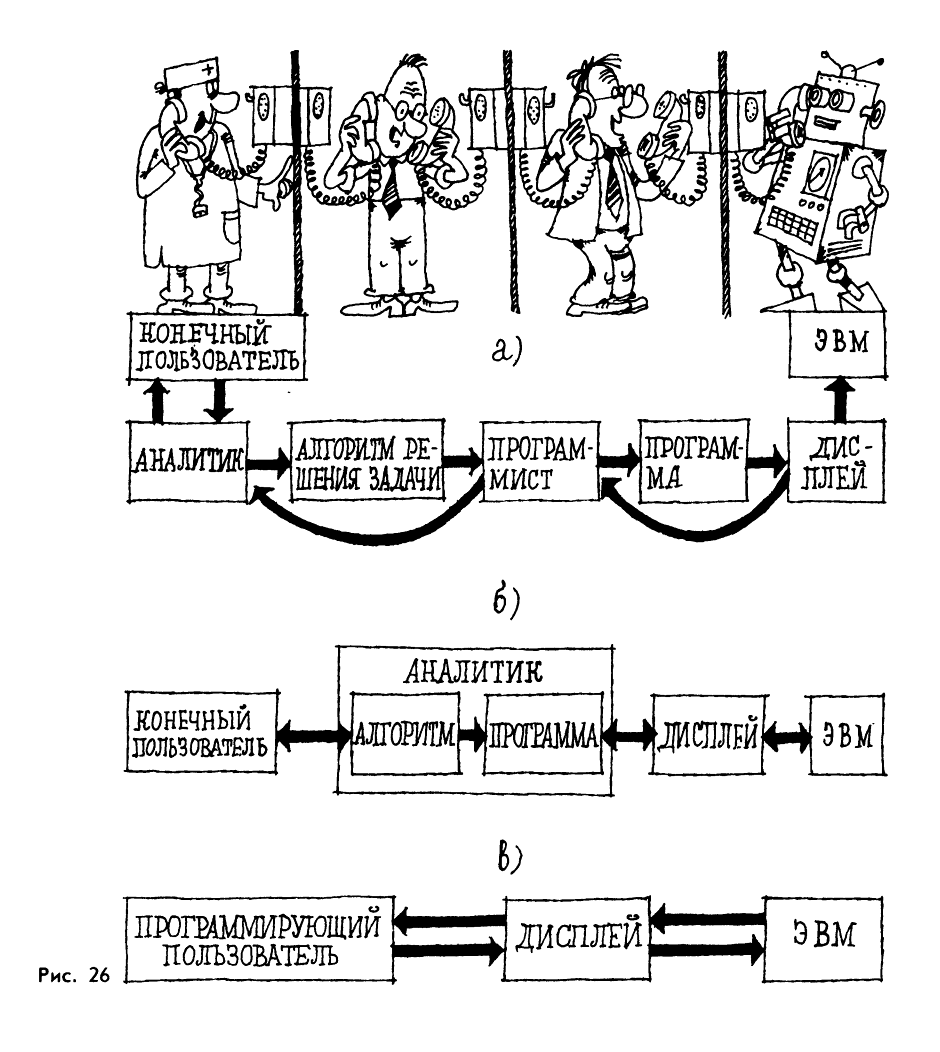


Рис. 26

для повышения эффективности его труда. Конечный пользователь и без нее может решить свою задачу, ведь справлялся же он с ней без компьютера! К компьютеру он обращается, чтобы решить свою задачу быстрее и лучше, чем прежде. Для него это лишь инструмент, позволяющий повысить свою производительность при решении конкретной задачи — решить ее или быстрее, или лучше, или и то и другое.

Как же быть? Многолетний опыт использования ЭВМ выработал схему, показанную на рис. 26, *а*. Здесь между пользователем и компьютером имеется по крайней мере два (а может быть, и больше) промежуточных звена.

ЗВЕНЬЯ ИСПОРЧЕННОГО ТЕЛЕФОНА

Первое звено — *аналитик* — это специалист по формальным методам решения задачи, т. е. чаще всего прикладной математик, знакомый с предметной областью пользователя. Это последнее свойство аналитика очень важно, иначе он просто не понимал бы конечного пользователя. Задачу пользователя, сформулированную ему на естественном языке, аналитик представляет в математической форме и разрабатывает (или заимствует из справочника) алгоритм ее решения. Например, радиолюбителю аналитик может составить уравнения, описывающие работу его схемы, и выбрать способ (алгоритм) их решения, чтобы определить характеристики этой схемы. Технологу аналитик составит модель, описывающую его технологический процесс, например, в виде системы массового обслуживания, которую можно моделировать с помощью ЭВМ, и т. д. Формализованную задачу и алгоритм ее решения аналитик передает программисту.

Второе звено — *программист* — составляет программу решения на машинном языке, вводит ее в ЭВМ и отлаживает ее. ЭВМ решает задачу и сообщает результат программисту, он расшифровывает его и передает аналитику, который переводит решение на язык пользователя.

Это простейшая схема решения задачи конечного пользователя на компьютере. Здесь между конечным пользователем и ЭВМ стоят два человека: аналитик и программист. Естественно желание устранить эти инстанции. Во-первых, эта схема очень похожа на игру в испорченный телефон, а во-вторых, аналитик и программист в ней участвуют почти всегда очень неохотно. Для них это, скорее, общественная нагрузка. Действительно, аналитик предпочитает разрабатывать методы решения задач, а не искать готовые рецепты. Программист также хочет заниматься своим делом — системным программированием (разработкой и эксплуатацией трансляторов, операционных систем и т. д.), а не программированием чужих задач.

СОКРАЩЕНИЕ ШТАТОВ

Первым шагом к решению этой проблемы было изобретение языков высокого уровня (Бейсик, Фортран, Алгол, GPSS и др., о них мы говорили в гл. 5), которые позволяют упростить процесс программирования. Например, для расчетов характеристик электронных схем обычно пользуются Фортраном, а для моделирования поведения систем массового обслуживания — одним из языков моделирования, например GPSS. Аналитик, владеющий одним из этих языков, уже не нуждается в программисте и может сам составить программу для решения задачи пользователя. Схема упрощается (рис. 26, *б*). Теперь самым «узким местом» в ней стал

только аналитик, выполняющий очень важную функцию формализации задачи пользователя. Чтобы убрать и эту инстанцию, можно пойти одним из двух путей:

передать функции аналитика конечному пользователю, чтобы он сам формализовал и программировал свою задачу.

передать эти функции ЭВМ, т. е. автоматизировать процесс формализации решения задачи пользователя и ее программирования.

Первый путь значительно проще: он не требует никаких дополнительных затрат, кроме ... обучения пользователя. Он должен научиться формализации (взять на себя функции аналитика) и научиться программировать (программиста). Необходима его двойная переквалификация — на прикладного математика в своей области и прикладного программиста, которых он должен подменить. Именно так приходится поступать сейчас каждому конечному пользователю, если он хочет решать свою задачу на ЭВМ. Почти все специалисты, использующие компьютер в своей области, проходят этот тернистый путь, требующий много времени, усидчивости, и становятся ... программирующими пользователями (рис. 26, *в*).

Поэтому так трудно внедрять компьютер в новые области. Чтобы использовать ЭВМ, кроме желания, надо еще очень много знать о методах формализации и решения формальных задач, а также о программировании, что, естественно, никак не может вдохновить. Действительно, эффективность специалиста при этом не повышается, а вполне реально снижается: ему приходится тратить много дополнительного времени на формализацию своей задачи и программирование, не говоря уж об отладке программы и других неизбежных программистских заботах. Именно поэтому многие специалисты отказываются применять компьютер в своей области, считая, что на их век хватит старых способов исследования, выработанных в докомпьютерную эпоху. Со многими из них можно согласиться.

Остается один путь — научить компьютер делать все самому: и понимать пользователя, и формализовать его задачу и программировать ее, и представлять результат ее решения в виде, доступном пониманию пользователя. Здесь компьютер должен выступать в очень сложной роли: «и швец, и жнец, и на дуде игрец». Таким умельцем (а точнее, интеллектуалом) должен быть компьютер, только так он даст возможность пользователю повысить эффективность его работы над своей профессиональной задачей, не отрывая его от этой задачи. Создание таких интеллектуальных экспертных систем в любой предметной области — дело чрезвычайно сложное.

Естественно возникает вопрос: а не слишком ли дорого мы платим за нежелание конечного пользователя самому разобраться в своей задаче, формализовать ее, запрограммировать и

т. д.? Если бы таких пользователей было бы немного, то, разумеется, «овчинка не стоила бы выделки». Создание таких интеллектуальных систем, точнее такой интеллектуальной программы, чрезвычайно трудоемкое дело. Но конечных пользователей, нуждающихся в интеллектуальных компьютерных системах, «тьмы и тьмы». Без большого преувеличения можно утверждать, что это все профессионалы, за исключением профессиональных программистов, а их не так уж много.

Не поможет ли компьютерная грамотность в решении этой проблемы? Хотя компьютерная грамотность и подразумевает знакомство с компьютером и одним из алгоритмических языков, но от знакомства до профессионального использования языка программирования — дистанция огромного размера. Знакомство, как известно, подразумевает лишь представление о возможностях компьютера. Но чтобы использовать все его возможности, надо создавать сложные программы, решающие поставленные задачи. И чем сложнее задача, тем труднее программировать ее решение — здесь надо профессионально знать все тонкости программирования.

Именно эти соображения привели к очень грустной мысли: в недалеком будущем всем нам, независимо от специальности, придется быть программистами и в кадровой анкете к графе о знании иностранных языков добавится еще одна — знание языка программирования. Но, к счастью, появление экспертных систем гарантирует нам возможность воспользоваться всей мощью современного компьютера без необходимости изучать его устройство и программирование.

ЧТО ДОЛЖНА УМЕТЬ ЭКСПЕРТНАЯ СИСТЕМА

Что же такое экспертная система (далее ЭС)? Внешне это компьютерная система, предназначенная для диалогового общения с непрограммирующим конечным пользователем. Этот пользователь ведет диалог с ЭС только на естественном языке. В процессе такого диалога ЭС «понимает» задачу пользователя, формализует ее, составляет программу решения, решает и выдает результат пользователю, причем полученные решения, как показывает опыт, бывают не только не хуже, а очень часто и лучше решений, принимаемых отдельными экспертами в этой области. Именно поэтому такие компьютерные системы назвали экспертными. ЭС — это первые компьютерные системы, которые доказали, что могут быть умнее человека в его области. Как же устроены ЭС?

Для своего нормального функционирования ЭС должна взять на себя функции аналитика в «старой» схеме (рис. 26, *б*), т. е. она должна:

1. Понимать естественный язык, на котором пользователь излагает свою задачу.

2. Уметь построить формальную модель этой задачи, чтобы применить формальные методы решения (ведь сила компьютера — в использовании формальных методов решения задач пользователя).

3. Составить программу решения задачи (или в простейшем случае найти эту программу в своем архиве — банке данных).

4. Запустить программу и получить результат.

5. Интерпретировать результат — представить его в форме, доступной и понятной пользователю.

6. Объяснить (при необходимости), как был получен этот результат.

Из этих шести пунктов только четвертый (прогон программы) традиционный, остальные же имеют прямое отношение к искусственному интеллекту. Рассмотрим основные из них.

Диалог на естественном языке

Понимание естественного языка является обязательной чертой всякой ЭС. При этом текст в компьютер может вводиться по-разному: с пульта дисплея или голосом через микрофон. Сам компьютер также может общаться с пользователем, выводя текст на экран дисплея или через синтезатор речи, который будет аккуратно произносить текст, синтезированный соответствующей программой. При таком общении пользователя с компьютером неизбежны моменты непонимания (как и между людьми). Например, в известной фразе «Я встретил ее на поле с цветами» совершенно непонятно, где цветы — на поле, у нее или у меня. Для выяснения такого рода недоразумений, чтобы правильно понимать пользователя, собеседник (в данном случае компьютер), должен уметь задавать вопросы. Таким образом, в процессе формулирования задачи между пользователем и ЭС должен происходить оживленный диалог, во время которого содержание задачи пользователя сообщается компьютеру. Программу, осуществляющую эту сложную операцию, называют *лингвистическим* процессором или (чаще) *диалоговым процессором,* подчеркивая важность диалогового характера процесса взаимодействия с пользователем.

База знаний

В своей работе диалоговый процессор активно взаимодействует с базой знаний, где хранятся знания из той предметной области, по которой специализирована данная ЭС. Сразу отметим, что нет ЭС на все случаи жизни, каждая ЭС довольно узко специализирована на определенную предметную область, например диагностика определенного вида заболеваний, проектирование систем заданного класса, поиск месторождений опреде-

ленного минерала и т. д. Узость предметной области ЭС позволяет создавать весьма полную базу знаний в этой области, что обеспечивает компьютеру возможность понимать пользователя «с полуслова». Но база знаний позволяет не только понимать пользователя, но и отвечать на его вопросы. Для этого она содержит сведения о том, как поступали раньше специалисты в той или иной ситуации и что из этого вышло. Эти знания представлены в виде так называемых *продукций* — конструкций вида «если ..., то ...». Например, в медицинской ЭС содержатся продукции вида «если больной имеет повышенную температуру и насморк, то это, возможно, грипп»; в геологоразведочной ЭС «если тип породы неизвестен, то надо проверить, какой из трех случаев имеет место: 1 — порода рыхлая, сыпучая; 2 — прочно связанная; 3 — легко ломается руками» и т. д. Особенно много ЭС специализируется по выяснению неисправностей в действующих системах. Их база знаний состоит из продукций вида «если характеристика А не в норме, то следует осмотреть блоки Б, В и Г» и т. д. С помощью таких экспертных знаний можно быстро найти неисправности в сложной технической системе. Опыт экспертов, отраженный в виде соответствующих продукций базы знаний, помогает отыскать неисправность, если с похожей неисправностью сталкивались эксперты, чьи знания содержатся в базе знаний.

Именно такого рода знания в базе дают возможность формализовать задачу пользователя, т. е. составить причинно-следственную цепочку, звеньями которой являются продукции из базы знаний, что в ее конце или будет дан ответ на заданный пользователем вопрос, или поставлен другой вопрос, на который нужно ответить пользователю.

Планирование

И, наконец, для эффективной работы ЭС необходимо превращать описание исходной задачи в рабочую программу, которая ее решает. Эту функцию выполняет *планировщик* — программная система, планирующая и организующая процесс решения поставленной задачи на ЭВМ. Планировщик постоянно контактирует с базой знаний, откуда черпает информацию и о способах решения той или иной задачи, и о том, как составляются рабочие программы для ЭВМ.

Как видно, основу ЭС составляют диалоговый процессор, база знаний и планировщик, которые и образуют интеллектуальный интерфейс между пользователями и компьютером (рис. 27). Его интеллектуальность определяется тем, что он понимает пользователя, формализует его задачу, составляет программу ее решения для компьютера и интерпретирует полученный результат (как говорится, «яичко испечет, да сам и облупит»). Едва ли

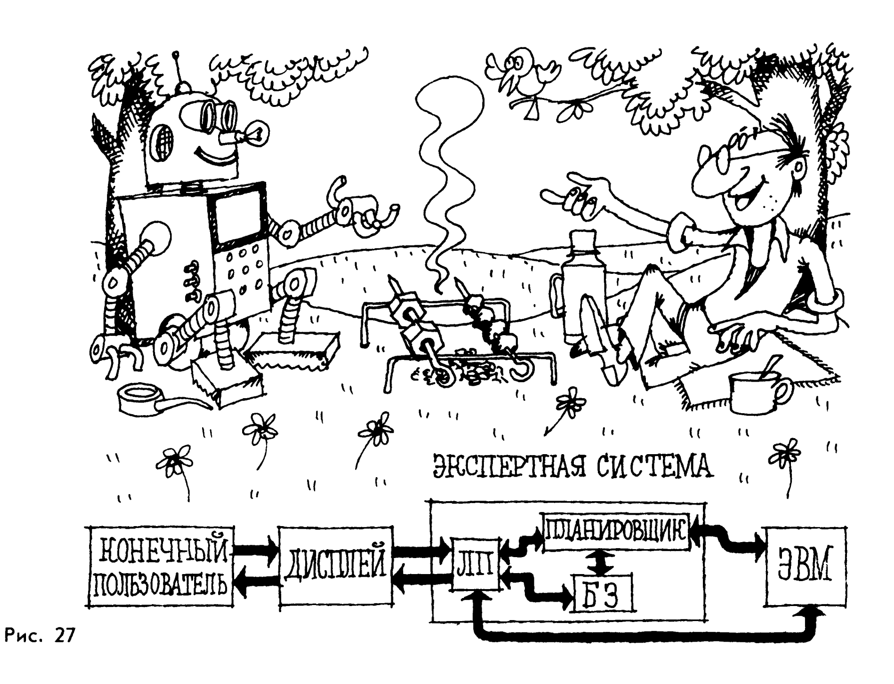


Рис. 27

у кого возникнут сомнения в интеллектуальности системы, совершающей такие действия.

Подсистема объяснения

И, наконец, последняя отличительная черта ЭС. Каждая ЭС имеет подсистему объяснения, которая позволяет, если необходимо, разъяснить пользователю, как ЭС получила то или иное решение. При этом широко используются сведения из базы знаний, с которыми согласен пользователь. Именно они являются аргументами, которые приводит ЭС, объясняя свое решение. Наличие такого объяснения делает полученное решение понятным и «прозрачным» пользователю. (Известно, что человек плохо воспринимает, а иногда и просто отвергает необоснованные советы. В этом проявляется критицизм — важная черта, защищающая его от слепого подчинения, свойственного, например, роботам, да и то не всем.)

Сейчас в мире уже существуют тысячи ЭС в самых разнообразных областях: медицине, технике, технологии, проектировании, геологоразведке, химии, экономике, юриспруденции и т. д. Эти области отличаются либо малой формализованностью, либо большой важностью экспертных решений, как, например, в технике, проектировании, экономике, и поэтому ценностью экспертных знаний, которые закладываются в базу знаний. Именно это

обстоятельство позволяет специалисту средней квалификации с помощью ЭС решать задачи, требующие высокой квалификации.

— Действительно, экспертные системы — очень удобная штука. А есть ли экспертная система по розыску преступников? Я бы с удовольствием пообщался с ней.

— Нет, и не может быть в принципе,— ответил Поль.— Ведь поимка преступников — это не узкая, а очень широкая область. А экспертные системы можно создавать лишь в крайне узких областях знаний и умений. Чтобы создать розыскную экспертную систему, прежде всего, надо выявить узкие классы преступления и наличие экспертов по этим преступлениям.

— Ну, что ж, такие эксперты существуют. Я знаю немало отменных специалистов-сыщиков, профилирующихся по отдельным видам преступлений: шантажам, мошенничествам, наркобизнесу, домашним кражам, угону автомобилей и т. д. Все они имеют огромный опыт раскрытия преступлений в своей области и могут выступить в роли экспертов. Но вопрос не в этом. Можно ли создать такую базу знаний? Ведь существование экспертов еще не гарантирует возможность построения экспертной системы. Не так ли, Поль?

— Вы правы, шеф. Кроме экспертов, нужно, чтобы область была очень узкой, только тогда база знаний будет обозримой. А это очень важно.

— Вот тебе и на!— удивился Мегрэ.— Неужели от компьютерной памяти зависит создание баз знаний?!

— Вовсе нет,— заторопился Поль,— на это память не экономят, хотя она и дорого стоит. Создание больших баз знаний требует больших и длительных усилий специалистов высокого класса — экспертов и инженеров по знаниям. Говоря проще, большие базы знаний стоят очень дорого.

— Ну и что? Это, так сказать, капитальные затраты на создание системы, которые быстро окупаются, если система интенсивно эксплуатируется. А недостатка в интересе к экспертным системам такого рода ощущаться не будет. Так что ваш аргумент против баз знаний в широкой области не очень убедителен.

— Есть и еще несколько неприятных обстоятельств,— уныло продолжал Поль.— Экспертная система для раскрытия преступлений должна обладать, кроме всего прочего, эрудицией, здравым смыслом и юридическими знаниями. Ведь именно эти качества должен иметь хороший следователь.

— Ну и давайте введем их в базу знаний,— бодро сказал Мегрэ.— Я даже слышал, что законы уже ввели в память компьютера и широко пользуются этой юридической базой знаний.

— Не знаний, а данных,— поправил Поль.— Ведь введение текста в память компьютера еще не создает базу знаний, это лишь данные. А знания образуются при введении смысла, т. е. при определении связей между понятиями, наполняющими базу данных. Базу знаний нельзя создать формально. Для этого всегда нужен высококвалифицированный специалист-эксперт, обладающий нужным знанием.

— Ну, а эрудиция и здравый смысл?— уже без энтузиазма спросил Мегрэ.— Их-то почему нельзя ввести в базу знаний?

— И то, и другое не является узкой областью знаний. Здравый смысл еще и не понятен сам по себе, а эрудиция требует огромных знаний, а не данных (которые

легко ввести в память компьютера, например, записав в его память текст многотомной энциклопедии).

— Ну, Поль,— обиженно сказал Мегрэ,— вот и вы заговорили о том, что недоступно компьютеру. А я-то уж было поверил в его неограниченные возможности и чуть было не стал крайне левым в вашей компьютерной братии.

— И очень жаль, шеф,— улыбнулся Поль.— Только компьютер — не прибор «для всего, чего изволите». Об этом уже много думали, а еще больше говорили. Ключевым словом здесь по-прежнему является «информация». Компьютер — это лишь машина для переработки информации, т. е. данных. С появлением необходимости компьютерного оперирования знаниями сразу появилось много ограничений на применение компьютера (о них уже говорилось выше). Этих ограничений будет еще больше, когда мы начнем оперировать компьютером в области искусства, этики, морали и т. д. И уж совсем трудно работать компьютеру с категориями сознания, сверхсознания, веры, совести и т. п.— очень человеческими категориями.

— Так что, так никогда они не будут доступны компьютеру? А может быть, это и хорошо, что у человека останется что-то его, чисто человеческое и недоступное машине?— с надеждой спросил Мегрэ.

— К сожалению, сказать это наверняка очень трудно. За непродолжительную, но бурную историю компьютера (ведь ему немногим более 40 лет) было немало прогнозов. Но все (почти все) они оказались несостоятельными. Вполне возможно, что в недалеком будущем будут найдены пути решения и тех задач, которые мы сейчас не знаем, как решить с помощью компьютера.

У физиков есть великолепное наблюдение: за каждой новой цифрой точности измерения скрывается открытие. С компьютером примерно то же. Каждый новый шаг повышения его производительности и емкости оперативной памяти открывает новые, ранее недоступные возможности его применения. Так, экспертные системы появились тогда, когда доступными стали компьютеры с миллионной оперативной памятью и соответствующей производительностью.

— Что ж, будем надеяться, что когда-нибудь компьютер будет консультировать и сыщика по самым сложным поворотам его дела, а не будет лишь компьютерным справочником или средством коммутации с банками данных.

БЕЗНАДЕЖНОЕ ДЕЛО ПРОГНОЗОВ

Так уж сложилось, что в заключение автор обязан дать прогноз в своей области, хотя бы на ближайшее будущее. Всякий прогноз рискован (вспомним печальную славу всех предсказателей), и тем более рискован, если относится к области, которая так бурно развивается, как вычислительная техника. Но тем не менее не будем нарушать традицию и попробуем заглянуть хотя бы в ближайшее будущее (о далеком прогнозе здесь и мечтать нельзя).

Начнем с персональной вычислительной техники — персональных компьютеров. Микроэлектронная технология развивается так интенсивно, что стоимость персональных компьютеров уменьшается, а объем производства увеличивается чрезвычайно быстро.

На Западе профессиональный персональный компьютер стоит уже приблизительно столько, сколько цветной телевизор, а бытовой — уже значительно ниже. Автор почти не рискует, утверждая, что в ближайшие пять лет персональные компьютеры у нас будут доступны всем нуждающимся.

Несколько хуже обстоит дело с подключением персонального компьютера к вычислительным сетям, обладающим большой вычислительной мощностью и информационными ресурсами. Здесь быстрый темп развития вычислительной техники сдерживается достаточно скромным ростом средств связи, необходимых для вычислительных сетей. Но к началу нового тысячелетия компьютеры пятого поколения должны стать основным средством переработки информации человечества. И именно это стимулирует развитие средств связи компьютеров. Только наличие каналов связи может обеспечить оперативное наращивание вычислительной мощности и информационных ресурсов, без чего невозможен дальнейший прогресс вычислительной техники.

А теперь о самом главном — о диалоге с компьютером. Эффективность его прямо зависит от возможностей компьютера, а они еще не столь велики, чтобы, например, вести диалог голосом. Свободный диалог, по-видимому, можно будет вести только с компьютерами пятого поколения, не раньше, чем лет через 10. Так что пока придется запастись терпением и овладевать техникой общения с компьютером через клавишный пульт. Это, разумеется, не исключает диалог голосом, но на усеченном языке и на узкую профессиональную тему. Причем «усеченность» и «узость» диалога будут постепенно сниматься.

И последнее. Не следует надеяться, что развитие вычислительной техники как-то кардинально изменит наше существование. Компьютер не более (но и не менее) чем один из мощных двигателей прогресса (как энергетика, металлургия, химия, машиностроение и т. д.), который берет на свои «железные плечи» такую важную функцию, как рутину обработки информации. Эта рутина всегда и везде сопровождает самые высокие полеты человеческой мысли. Именно в этой рутине очень часто тонут дерзкие решения, недоступные компьютеру. Поэтому так важно «свалить» на компьютер рутинные операции, чтобы освободить человека для его истинного предназначения — творчества.

Вспомним знаменитые слова М. Горького «Все — в человеке, все для человека! Существует только человек, все же остальное — дело его рук и его мозга». Компьютер — тоже дело рук и мозга человека.

Этим гимном человеку и его необыкновенной способности творить мы и закончим книгу о взаимоотношении человека-творца и компьютера, пожалуй, самого активного помощника человека в его творчестве.

ДЕТЕКТИВ ВМЕСТО ПОСЛЕСЛОВИЯ

— Комиссар,— воскликнул Поль,— что вы нашли в комнате погибшей? Какое такое ЭТО; Дело прошлое, и пора вам раскрыть свои карты!

— Да, признаюсь, я сам не ожидал напасть на такой след,— довольно улыбнулся Мегрэ,— и, хотя мне так и не удалось его довести до конца, кое-что все-таки получилось.

— Да не темните, шеф!— нетерпеливо прервал его Поль.— Что такое ЭТО! Или вы решили унести эту тайну с собой?

— Зачем же!— и Мегрэ стал раскуривать свою трубку.— Думаю, что вам будет полезно узнать, что такое ЭТО.

— Не торопитесь, дружище. Согласно всем правилам детективного жанра я должен издалека и очень обстоятельно подвести вас к отгадке, которую вы так хотите узнать,— и Мегрэ лукаво взглянул на Поля.

— Я подозреваю, шеф,— обиженно произнес Поль,— что и вы не знаете, что такое ЭТО. Я с самого начала догадывался, что это дешевый трюк автора, на который должен клюнуть неискушенный читатель и начать читать эту книгу. А ЭТО — пирожок ни с чем! Я угадал? Держу какое угодно пари, что вы сами не знаете, что такое ЭТО!

— Вы уже проиграли его! У ЭТОГО есть название, и оно известно тем, кто читал польского фантаста Станислава Лема.

— Ну, я читал!— сказал Поль.— У меня к Лему отношение двойственное: наполовину он великолепный фантаст, а вот другой половиной он просто насмешник. Его путешествия Ийона Тихого — это издевательство над фантастикой!

— Вы просто зануда, Поль,— захохотал Мегрэ.— Ведь эти путешествия — великолепная пародия на современную фанстастику. Причем пародия многослойная, и очень острая.

— Не отвлекайтесь, шеф. Вы обещали назвать ЭТО!

— Это — *сепулька**,— спокойно сказал Мегрэ и пустил кольцо дыма в потолок.

— Что, что!!!— остолбенел Поль.

— Да, да; Обыкновенная *сепулька*. Та самая, о которой писал Лем в одном из путешествий Ийона Тихого, кажется в четырнадцатом.

— Ну, и какая же она?— пришел в себя Поль.

— Именно такая, как описал ее Лем. Вы что, не помните?

— Я все хорошо помню. Даже помню его издевательское описание: *«Сепулька* — элемент цивилизации *ардритов,* См. *сепулькарии» «Сепулькарии* — устройства, служащие для *сепуления* (см)», *«Сепуление* — занятие *ардритов,* См. *Сепулька»*. Что вы еще можете добавить к этому?

— Все правильно,— спокойно улыбнулся Мегрэ.— Добавлю лишь то, что *сепулька* сейчас стала объектом подпольного бизнеса у нас на Земле. И международная мафия начинает переходить с наркобизнеса к сепбизнесу — бизнесу на *сепульках*. Именно ее я увидел в комнате погибшей.

— На что же она похожа, эта *сепулька?*— тихо спросил Поль.

---

* Здесь и ниже курсивом выделены фантастические термины и понятия, введенные С. Лемом.

— Ну, это была не обычная *сепулька,* а с *присвистом,* которую используют для торжественных *сепулений.* К сожалению, у меня не было *щерси,* чтобы *закруситься.* Но зато был резерв, который начал *бобчить,* и мне ничего не оставалось, как обратиться к *ардриту.*

— Но ведь это житель планеты *Интеропии!*— заикаясь произнес Поль и начал *светиться.*

— Конечно; Как же без него я смог бы сделать *розмазь каймистости,* которая обязательна.— Мегрэ задумался и добавил,— А вы думаете, что можно было бы обойтись без нее? Я об этом не подумал! Что с вами, Поль? Уж если вы начали *светиться,* то наденьте абажур, как это принято у *ардритов!*

— Нет!— слабо простонал Поль,— я жду *хемпа,* чтобы воспользоваться своим *резервом.* Или *курдля.* И вообще, я хочу в *Интеропию.*— Он *засветился* еще ярче, *превратился в шар и покатился по потолку.*

— Ох уж, эта молодежь!— проворчал Мегрэ, провожая шар взглядом.— Вечно все принимают слишком близко к сердцу. И никак не поймут, что далеко не все следует понимать. Даже в вычислительной технике, будучи «наедине с компьютером»,— он усмехнулся и захлопнул книгу.

ЛЮБОЗНАТЕЛЬНОМУ ЧИТАТЕЛЮ

Автор ориентировался на некоего «среднего» по его мнению читателя, которого, как известно, не существует, а «нормально» любознательному читателю хотел бы посоветовать следующие книги.

Батурин Ю. М. Право и политика в компьютерном круге.— М.: Наука, 1987.—110 с.

Войскунский С. Я говорю...— М.: Знание, 1984.

Геловани В. А., Ковригин О. В. Экспертные системы в медицине.— М.: Знание, 1987.—32 с.— (Новое в жизни, науке, технике. Сер. Математика, кибернетика, № 3).

Горелов И. Н. Разговор с компьютером: Психолингвистический аспект проблемы.— М.: Наука, 1987.—256 с.

Горстко А. Б., Кочковая С. В. Азбука программирования.— М.: Знание, 1988.— 144 с.

Громов Г. Р. Национальные информационные ресурсы: Проблемы промышленной эксплуатации.— М.: Наука, 1984.

Евреинов Э. В. Распределенная обработка информации и распределенные вычислительные системы.— М.: Знание, 1983.—57 с.— (Новое в жизни, науке, технике. Сер. Радиоэлектроника и связь, № 3).

Журавлев А. Диалог с компьютером.— М.: Молодая гвардия.—1988.

Иоффе А. Ф. Персональные ЭВМ в организационном управлении.— М.: Наука, 1988.—208 с.

Кондратов А. Электронный разум: Очерк исследований по проблеме искусственного интеллекта.— М.: Знание, 1987.—176 с.

Микроэлектронная технология и ее влияние на общество: Пер. с англ.— М.: Знание, 1987.—160 с.

Мичи Д., Джонстон Р. Компьютер — творец: Пер. с англ.— М.: Мир, 1987.

Перегудов М. А., Халмейзер А. Я. Бок о бок с компьютером.— М.: Высшая школа, 1987.—192 с.

Персональные компьютеры.— М.: Наука, 1987.—149 с.— (Информатика для всех).

Персональный компьютер в играх и задачах.— М.: Наука, 1988.—192 с.— (Сер. Кибернетика — неограниченные возможности и возможные ограничения).

Печерский Ю. Н. На пути к искусственному интеллекту.— Кишинев: Штиница, 1981.—182 с.

Попов Э. В. Экспертные системы. Решение неформализованных задач в диалоге с ЭВМ.— М.: Наука, 1987.—288 с.

Поспелов Г. С., Поспелов Д. А. Искусственный интеллект — прикладные системы.— М.: Знание, 1985.—48 с.

Поспелов Д. А. Большие системы (ситуационное управление).— М., 1975.— 62 с.

Поспелов Д. А. Фантазия и наука. На пути к искусственному интеллекту.— М.: Наука, 1982.—217 с.

Построение экспертных систем: Пер. с англ.— М.: Мир, 1987.—441 с.

Растригин Л. А. Вычислительные машины, системы, сети...— М.: Наука, 1982.—224 с.

Реальность и прогнозы искусственного интеллекта: Пер. с англ. М.: Мир, 1987.— (В мире науки и техники).

Современный компьютер: Пер. с англ. под ред. В. М. Курочкина.—М.: Мир, 1986.—210 с.

Уинстон П. Искусственный интеллект: Пер. с англ.— М.: Мир, 1980.—520 с.

Фурунджиев Р. И., Бабушкин Ф. М., Варавко В. В. Диалог с ЭВМ.— Минск: Высшая школа. 1986.—160 с.— (Мир занимательной науки).

Хелмс Г. Л. Языки программирования. Краткое руководство: Пер. с англ.— М.: Радио и связь, 1985.

ЭВМ пятого поколения. Концепции, проблемы, перспективы.—М.: Финансы и статистика, 1984.—110 с.

Экспертные системы. Принципы работы и примеры: Пер. с англ. М.: Радио и связь, 1987.—224 с.

Элти Д., Кумбс М. Экспертные системы. Концепции и примеры: Пер. с англ.— М.: Финансы и статистика, 1987.—191 с.

СОДЕРЖАНИЕ

Научно-популярное издание

Массовая радиобиблиотека Выпуск № 1157
Растригин Леонард Андреевич

С КОМПЬЮТЕРОМ НАЕДИНЕ

Заведующая редакцией **Г. И. Козырева**
Редактор **Т. М. Толмачева**
Обложка и рисунки художника **К. Н. Мошкина**
Художественный редактор **Н. С. Шеин**
Технический редактор **Т. Н. Зыкина**
Корректр **О. Е. Иваницкая**

ИБ № 2091

Сдано в набор 12.04.89. Подписано в печать 25.07 89. Т-23766. Формат 60×88 1/16. Бумага тип № 2. Гарнитура литер. Печать офсет. Усл. п. л. 13,72 Усл. кр -отт. 14,21 Уч.-изд. л. 15,42. Тираж 100 000 экз. Изд. № 22328. Заказ 2074. Цена 1 р. 20 к.

Издательство «Радио и связь», 101000, Москва, Почтамт, а/я 693.
Московская типография № 4 Госкомпечати СССР.
Москва, И-41, Б. Переяславская, 46.

1 р. 20 к.



